Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 220 (2006) 109–138
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

An immersed interface method for viscous incompressible flows involving rigid and flexible boundaries

D.V. Le [a], B.C. Khoo [a,b,*], J. Peraire [a,c]

[a] Singapore-MIT Alliance, 4 Engineering Drive 3, National University of Singapore, Singapore 117576, Singapore
[b] Department of Mechanical Engineering, National University of Singapore, Kent Ridge Crescent, Singapore 119260, Singapore
[c] Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 77 Massachusetts Avenue, MA 02139, USA

Received 22 September 2005; received in revised form 5 May 2006; accepted 5 May 2006
Available online 13 July 2006

Abstract

We present an immersed interface method for the incompressible Navier–Stokes equations capable of handling both rigid and flexible boundaries. The immersed boundaries are represented by a number of Lagrangian control points. In order to ensure that the no-slip condition on the rigid boundary is satisfied, singular forces are applied on the fluid. The forces are related to the jumps in pressure and the jumps in the derivatives of both pressure and velocity, and are interpolated using cubic splines. The strength of the singular forces at the rigid boundary is determined by solving a small system of equations at each timestep. For flexible boundaries, the forces that the boundary exerts on the fluid are computed from the constitutive relation of the flexible boundary and are applied to the fluid through the jump conditions. The position of the flexible boundary is updated implicitly using a quasi-Newton method (BFGS) within each timestep. The Navier–Stokes equations are discretized on a staggered Cartesian grid by a second order accurate projection method for pressure and velocity and the overall scheme is second order accurate.

Keywords: Immersed interface method; Navier–Stokes equations; Cartesian grid method; Finite difference; Fast Poisson solvers; Irregular domains; Fluid–membrane interaction

1. Introduction

In this paper, we present a numerical method for solving viscous, incompressible flow problems involving both moving interfaces and rigid boundaries. One of the challenges in these problems is that the fluid motion, the flexible interface motion and the interaction with the immersed rigid boundaries must be computed simultaneously. This is necessary in order to account for the complex interaction between the fluid and the immersed boundaries. An example of interface problems that we consider is shown in Fig. 1. In a 2-dimensional bounded

* Corresponding author. Address: Department of Mechanical Engineering, National University of Singapore, Kent Ridge Crescent, Singapore 119260, Singapore. Tel.: +65 68742889; fax: +65 67791459.
E-mail address: mpekbc@nus.edu.sg (B.C. Khoo).

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2006.05.004

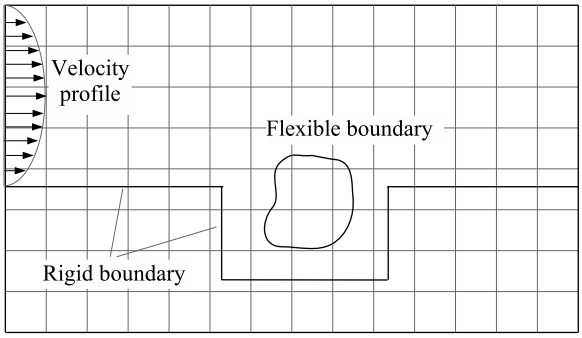


Fig. 1. A typical domain in which the Navier–Stokes equations are solved. The flexible interface and the rigid boundary are immersed in a uniform Cartesian grid.

domain Ω that contains a material interface $\Gamma(t)$, we consider the incompressible Navier–Stokes equations, written as

$$\rho(\boldsymbol{u}_t + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}) + \nabla p = \mu\Delta\boldsymbol{u} + \boldsymbol{F}, \tag{1}$$

$$\nabla\cdot\boldsymbol{u} = 0 \tag{2}$$

with boundary conditions

$$\boldsymbol{u}|_{\partial\Omega} = \boldsymbol{u}_b, \tag{3}$$

where $\boldsymbol{u}$ is the fluid velocity, p is the pressure, ρ is the density, and μ the viscosity of the fluid. Throughout this paper, we assume that the fluid density ρ and the viscosity μ are constant over the whole domain. The effect of the material interface $\Gamma(t)$ immersed in the fluid results in a singular force $\boldsymbol{F}$ which has the form

$$\boldsymbol{F}(\boldsymbol{x},t) = \int_{\Gamma(t)} \boldsymbol{f}(s,t)\delta(\boldsymbol{x}-\boldsymbol{X}(s,t))\,\mathrm{d}s, \tag{4}$$

where $\boldsymbol{X}(s,t)$ is the arc-length parametrization of $\Gamma(t)$, s is the arc-length, $\boldsymbol{x}=(x,y)$ is spatial position, and $\boldsymbol{f}(s,t)$ is the force strength. Here, $\delta(\boldsymbol{x})$ is the two-dimensional Dirac function. The motion of the interfaces satisfies

$$\frac{\partial}{\partial t}\boldsymbol{X}(s,t) = \boldsymbol{u}(\boldsymbol{X},t) = \int_{\Omega} \boldsymbol{u}(\boldsymbol{x},t)\delta(\boldsymbol{x}-\boldsymbol{X}(s,t))\,\mathrm{d}\boldsymbol{x}. \tag{5}$$

In our proposed numerical method, the Navier–Stokes equations are discretized using a standard finite difference method on a staggered Cartesian grid. Methods utilizing a Cartesian grid for solving interface problems or problems with complex geometry have become popular in recent years. Existing Cartesian grid methods for interface problems can be categorized into two general groups: methods that determine the jump conditions across the interface and incorporate them into the finite difference scheme and methods that smooth out the singular force before it is applied to the fluid. Our method which is based on the immersed interface method originally proposed by LeVeque and Li [20,21] falls into the first group. The immersed boundary method introduced by Peskin [26] belongs to the second group.

Peskin's immersed boundary method has proven to be a very useful method for modelling fluid-structure interaction involving large geometry variations. This method has been applied to many biological problems involving flexible boundaries [10,11,25,34]. In the immersed boundary method, the force densities are computed at the control points which are used to represent the boundaries. The force densities are then spread to the Cartesian grid points by a discrete representation of the delta function. The Navier–Stokes equations with the forcing terms are then solved for pressure and velocity at the Cartesian grid points. Further details on the immersed boundary method can be found in [26] and the references therein. The immersed boundary method has several attractive features: the method is simple to implement, it can handle complex geometries easily and it uses standard regular Cartesian grid Navier–Stokes solvers. However, since the immersed boundary method uses the discrete delta function approach, it smears out sharp interface to a thickness of

order of the meshwidth and it is only about first-order accurate for general problems. The immersed boundary method has also been applied to problems with rigid boundaries [15,31]. In order to deal with rigid boundaries, Lai and Peskin [15] evaluated the force density using a particular case of the feedback forcing formulation proposed by Goldstein et al. [13] so as to ensure that the boundary points will stay close to the required boundary position. An alternative model to compute the force density $\boldsymbol{f}$ was proposed in [31] based on the use of the momentum Eq. (1) at the boundaries. These forcing formulations are simple to implement but require a small timestep to maintain the stability.

In contrast, the immersed interface method (IIM) can avoid smearing sharp interfaces and maintains second-order accuracy by incorporating the known jumps into the finite difference scheme near the interface. The singular force $\boldsymbol{f}$ along the immersed boundaries results in solution to the Navier–Stokes equations which may be non-smooth across the interface, i.e., there may be jumps in pressure and in the derivatives of both pressure and velocity at the interface. An essential ingredient of the immersed interface method is the relation between the jumps in the solutions and their derivatives, and the applied singular forces. The basic idea of the immersed interface method is to discretize the Navier–Stokes equations on a uniform Cartesian grid and to account for the singular forces by explicitly incorporating the jumps in the solutions and their derivatives into the difference equations. The IIM was originally proposed by LeVeque and Li [20] for solving elliptic equations, and later extended to Stokes flow with elastic boundaries or surface tension [21]. The method was developed further for the Navier–Stokes equations in [17,19,22] for problems with flexible boundaries. In [22], the level set method is used to represent the interface. This has the advantage of simplifying the algorithm but does not appear to be adequate to represent certain types of interfaces such as elastic membranes. In [19], the interface is tracked explicitly in a Lagrangian manner, the singular force $\boldsymbol{f}$ is split into components tangential and normal to the interface. The normal component is then incorporated into jump conditions for pressure across the interface. The tangential component is spread to the nearby Cartesian grid points using the discrete delta function as in the immersed boundary method [26]. Spreading the tangential force to the nearby Cartesian grid points has the effect of smoothing out the jumps in the derivatives of pressure and velocity. The IIM was also used in [7,23,29] for solving the two-dimensional streamfunction-vorticity equations on irregular domains.

In the present work, we introduce a formulation of the immersed interface method for solving the incompressible Navier–Stokes equations in the presence of rigid boundaries. Our approach is largely based on that described in Le et al. [18]. In addition, we also combine this algorithm with our earlier work for problems with flexible boundaries [17] to handle rigid and flexible boundaries simultaneously. It may be noted that most of the current Cartesian grid methods can only handle flexible boundaries and the rigid boundaries are usually required to be aligned with the computational grid [2,28]. This is contrasted to our proposed method where arbitrary piecewise smooth rigid boundaries can be considered. Therein lies one main advantage of our method to simulate the motion of multiple deformable boundaries in a domain with multiple immersed rigid boundaries.

Our approach employs the immersed interface method to solve the incompressible Navier–Stokes equations formulated in primitive variables. The singular force at the rigid boundary is determined for imposition of the no-slip condition. At each time step the singular force is computed implicitly by solving a small, dense linear system of equations. In this way, we can impose exactly the no-slip condition at the boundary and avoid the need for very small timesteps. The singular force at the flexible interface is computed based on the configuration of the interface, i.e., the interface is assumed to be governed by either surface tension, or by an elastic membrane. Note that the entire singular force at the flexible interface is incorporated into the jump conditions for pressure and the derivatives of pressure and velocity. As such, our algorithm can successfully capture all the jumps in the solutions and their derivatives. Having computed the singular force, we next compute the jump in pressure and jumps in the derivatives of both pressure and velocity. The jumps in the solution and its derivatives are incorporated into the finite difference discretization to obtain a sharp interface resolution. Fast solvers from the FISHPACK software library [1] are used to solve the resulting discrete systems of equations.

The remainder of the paper is organized as follows. In Section 2, we present the relations that must be satisfied along the immersed boundary between the singular force $\boldsymbol{f}$ and the jumps in the velocity and pressure and their derivatives. In Section 3, we describe the generalized finite difference approximations to the solution derivatives, which incorporate the solution jumps. In Section 4, we present the numerical algorithm. In Section 5, some numerical examples are presented and finally, some conclusions and suggestions for future work are given in Section 6.

2. Jump conditions across the interface

We have already mentioned that when singular forces are applied on a material interface, the solutions of the Navier–Stokes equations may be non-smooth or discontinuous across the interface. Let $\boldsymbol{n}$ and $\boldsymbol{\tau}$ be the unit outward normal and tangential vectors to the interface, respectively. The respective normal and tangential components of the force density $f_1 = \boldsymbol{f}(s,t)\cdot\boldsymbol{n}$ and $f_2 = \boldsymbol{f}(s,t)\cdot\boldsymbol{\tau}$ can be related to the jump conditions for pressure and velocity as follows:

$$[\boldsymbol{u}] = \boldsymbol{0}, \quad [\mu \boldsymbol{u}_\xi] = -f_2\boldsymbol{\tau}, \quad [\boldsymbol{u}_\eta] = \boldsymbol{0}, \tag{6}$$

$$[p] = f_1, \quad [p_\xi] = \frac{\partial f_2}{\partial s}, \quad [p_\eta] = \frac{\partial f_1}{\partial s}, \tag{7}$$

$$[\mu \boldsymbol{u}_{\eta\eta}] = \kappa f_2 \boldsymbol{\tau}, \quad [\mu \boldsymbol{u}_{\xi\eta}] = -\frac{\partial f_2}{\partial \eta}\boldsymbol{\tau} - \kappa f_2 \boldsymbol{n},$$

$$[\mu \boldsymbol{u}_{\xi\xi}] = -[\mu \boldsymbol{u}_{\eta\eta}] + [p_\xi]\boldsymbol{n} + [p_\eta]\boldsymbol{\tau} + \rho[\boldsymbol{u}_\xi]\boldsymbol{u}\cdot\boldsymbol{n}. \tag{8}$$

The above equations were derived in [22] and here, we have used the same notation for clarity. The jump, $[\cdot]$, denotes the difference between the value of its argument outside and inside the interface, and (ξ,η) are the rectangular coordinates associated with the directions of $\boldsymbol{n}$ and $\boldsymbol{\tau}$, respectively. Here, κ is the signed valued of the curvature of the interface (i.e. we assume that $\boldsymbol{n}\times\boldsymbol{\tau} = \boldsymbol{k} \equiv$ constant, so that $\boldsymbol{n}$ can point either towards, or outwards from, the center of curvature). We note that from expressions (6)–(8) the values of the jumps of the first and second derivatives of velocity and pressure taken with respect to the (x,y) coordinates are easily obtained by a simple coordinate transformation. For instance, we have

$$[\boldsymbol{u}_x] = [\boldsymbol{u}_\xi]n_1 + [\boldsymbol{u}_\eta]\tau_1,$$
$$[\boldsymbol{u}_{yy}] = [\boldsymbol{u}_{\xi\xi}]n_2^2 + 2[\boldsymbol{u}_{\xi\eta}]n_2\tau_2 + [\boldsymbol{u}_{\xi\eta}]\tau_2^2,$$

where $\boldsymbol{n} = (n_1, n_2)$ and $\boldsymbol{\tau} = (\tau_1, \tau_2)$ are the Cartesian components of the normal and tangential vectors to the interface at the point considered.

3. Generalized finite difference formulas

From Taylor series expansions, it is possible to show that if the interface cuts a grid line between two grid points at $x = \alpha, x_i \leqslant \alpha < x_{i+1}, x_i \in \Omega^-, x_{i+1} \in \Omega^+$, then the following approximations hold for a piecewise twice differentiable function $v(x)$:

$$v_x(x_i) = \frac{v_{i+1} - v_{i-1}}{2h} - \frac{1}{2h}\sum_{m=0}^{2}\frac{(h^+)^m}{m!}[v^{(m)}] + \mathrm{O}(h^2) \tag{9}$$

$$v_x(x_{i+1}) = \frac{v_{i+2} - v_i}{2h} - \frac{1}{2h}\sum_{m=0}^{2}\frac{(h^-)^m}{m!}[v^{(m)}] + \mathrm{O}(h^2) \tag{10}$$

$$v_{xx}(x_i) = \frac{v_{i+1} - 2v_i + v_{i-1}}{h^2} - \frac{1}{h^2}\sum_{m=0}^{2}\frac{(h^+)^m}{m!}[v^{(m)}] + \mathrm{O}(h) \tag{11}$$

$$v_{xx}(x_{i+1}) = \frac{v_{i+2} - 2v_{i+1} + v_i}{h^2} + \frac{1}{h^2}\sum_{m=0}^{2}\frac{(h^-)^m}{m!}[v^{(m)}] + \mathrm{O}(h) \tag{12}$$

where $v^{(m)}$ denotes the mth derivative of $v, v_i = v(x_i), h^+ = x_{i+1} - \alpha$, $h^- = x_i - \alpha$ and h is the mesh width in x direction. The jumps in v and its derivatives are defined as

$$[v^{(m)}]_\alpha = \lim_{x\to\alpha, x\in\Omega^+} v^{(m)}(x) - \lim_{x\to\alpha, x\in\Omega^-} v^{(m)}(x) \tag{13}$$

in short, $[\cdot] = [\cdot]_\alpha$, and $v^{(0)} = v$. See Weigmann and Bube [35] for detailed proofs of expressions (9)–(12). Note that if the interface cuts a grid line between two grid points $x_i \in \Omega^+$ and $x_{i+1} \in \Omega^-$, these expressions need to

be modified by changing the sign of the second terms on the respective right-hand sides. Expressions involving two or more interface crossings could also be derived, see for example [35].

Finally, we also require centered and backwards approximations for $v(t^{n+1/2})$. These approximations come about when the interface crosses a grid point over the time interval considered. Thus, assuming that the interface crosses a grid point at time τ, we have the following approximations:

(a) centered

$$v(t^{n+1/2}) = \begin{cases} \frac{1}{2}(v^n + v^{n+1}) + \frac{1}{2}[v]_\tau + O(\Delta t), & t^n \leqslant \tau < t^{n+1/2} \\ \frac{1}{2}(v^n + v^{n+1}) - \frac{1}{2}[v]_\tau + O(\Delta t), & t^{n+1/2} \leqslant \tau < t^{n+1} \end{cases} \tag{14}$$

(b) backwards

$$v(t^{n+1/2}) = \begin{cases} \frac{3}{2}v^n - \frac{1}{2}v^{n-1} - \frac{1}{2}[v]_\tau + O(\Delta t), & t^{n-1} \leqslant \tau < t^n \\ \frac{3}{2}v^n - \frac{1}{2}v^{n-1} + [v]_\tau + O(\Delta t), & t^n \leqslant \tau < t^{n+1/2}. \end{cases} \tag{15}$$

Here, $[v]_\tau$ denotes the jump in time of a function $v(x,t)$ at a particular grid point and is only non zero when the interface crosses the grid point at time τ. The jump in time is defined as

$$[v(t)]_\tau = \lim_{t\to\tau^+} v(t) - \lim_{t\to\tau^-} v(t). \tag{16}$$

It is easy to see that $[v]_\alpha = \pm[v]_\tau$, where $[\cdot]_\alpha$ denotes spatial jump as defined in (13) and the sign depends on the motion of the interface. In particular, we use a plus sign when the grid point moves from the inside of the interface to the outside of the interface, i.e. from Ω^- to Ω^+, and a minus sign when the grid point moves from the outside of the interface to the inside of the interface, i.e. from Ω^+ to Ω^-.

4. Numerical algorithm

4.1. Projection method

We employ a pressure-increment projection algorithm for the discretization of the Navier–Stokes equations. This projection algorithm is analogous to that presented in Brown et al. [5]. It leads to a second order accuracy for both velocity and pressure provided all the spatial derivatives are approximated to second order accuracy. The spatial discretization is carried out on a standard marker-and-cell (MAC) staggered grid similar to that found in Kim et al. [14]. The ENO third-order upwind scheme is used for the advective terms [30]. With the MAC mesh, the pressure field is defined at the cell center where the continuity equation is enforced. The velocity fields u and v are defined at the vertical edges and horizontal edges of a cell, respectively. One advantage of the MAC mesh is that boundary conditions for pressure are not required explicitly. On the other hand, the use of a non-staggered grids introduces some complications. For instance, some of the velocity components are not defined on the boundaries of the domain.

The pressure-increment procedure for problems with immersed interfaces is the same as that for non-interface problems. For problems with immersed interface, however, the discretization of the Navier–Stokes equations at those grid points near the interface needs to be modified to account for the jump conditions across the interface. Below, we review the pressure-increment method for the case of immersed interfaces. Given the velocity $\boldsymbol{u}^n$, and the pressure $p^{n-1/2}$, we compute the velocity $\boldsymbol{u}^{n+1}$ and pressure $p^{n+1/2}$ at the next time step in three steps:

Step 1: Compute an intermediate velocity field $\boldsymbol{u}^*$ by solving

$$\begin{aligned} &\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} = -\overline{(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^{n+\frac{1}{2}}} - \frac{1}{\rho}\overline{\nabla p^{n+\frac{1}{2}}} + \frac{\mu}{\rho}\overline{\nabla^2\boldsymbol{u}^{n+\frac{1}{2}}} + \boldsymbol{C}_1 \\ &\boldsymbol{u}^*|_{\partial\Omega} = \boldsymbol{u}_b^{n+1} \end{aligned} \tag{17}$$

where the advective term is extrapolated using the formula,

$$\overline{(\boldsymbol{u}.\nabla\boldsymbol{u})^{n+\frac{1}{2}}} = \frac{3}{2}(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^{n} - \frac{1}{2}(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^{n-1} + \boldsymbol{C}_2 + \gamma_1[\boldsymbol{u}\cdot\nabla\boldsymbol{u}]_\tau, \tag{18}$$

the diffusion term is approximated implicitly as

$$\overline{\nabla^2\boldsymbol{u}^{n+1/2}} = \frac{1}{2}(\nabla_h^2\boldsymbol{u}^* + \nabla_h^2\boldsymbol{u}^n) + \boldsymbol{C}_3 + \gamma_2[\nabla_h^2\boldsymbol{u}]_\tau, \tag{19}$$

and the pressure gradient is approximated simply as

$$\overline{\nabla p^{n+\frac{1}{2}}} = G^{\mathrm{MAC}}p^{n-\frac{1}{2}} + \boldsymbol{C}_4 + \gamma_3[\nabla p]_\tau. \tag{20}$$

The MAC gradient operators are defined as

$$(G_x^{\mathrm{MAC}}p)_{i+\frac{1}{2},j} = \frac{p_{i+1,j}-p_{i,j}}{\Delta x}, \quad (G_y^{\mathrm{MAC}}p)_{i,j+\frac{1}{2}} = \frac{p_{i,j+1}-p_{i,j}}{\Delta y}$$

Step 2: Compute a pressure update ϕ^{n+1} by solving the Poisson equation

$$\nabla_h^2\phi^{n+1} = \rho\frac{D^{\mathrm{MAC}}\boldsymbol{u}^*}{\Delta t} + C_5, \tag{21}$$

with boundary condition

$$\boldsymbol{n}\cdot\nabla\phi^{n+1}|_{\partial\Omega} = 0. \tag{22}$$

The MAC divergence operator is defined as

$$(D^{\mathrm{MAC}}\boldsymbol{u})_{i,j} = \frac{u_{i+\frac{1}{2},j}-u_{i-\frac{1}{2},j}}{\Delta x} + \frac{v_{i,j+\frac{1}{2}}-v_{i,j-\frac{1}{2}}}{\Delta y}.$$

Step 3: Update pressure and velocity field

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \frac{1}{\rho}\Delta t G^{\mathrm{MAC}}\phi^{n+1} + \boldsymbol{C}_6 \tag{23}$$

$$p^{n+1/2} = p^{n-1/2} + \phi^{n+1} - \frac{\mu}{2\rho}(D^{\mathrm{MAC}}\boldsymbol{u}^*) + C_7 \tag{24}$$

Here, $[\cdot]_\tau$ denotes a jump in time and is only non zero when the interface crosses the grid point over the time interval considered. The coefficients γ_i, $i = 1, 2, 3$ correspond to the first order corrections in time. The coefficient γ_1 is determined from expression (15) and the coefficient γ_2 is determined from expression (14). The coefficient γ_3 is only nonzero when the interface crosses the grid point over the time interval $[t^{n-1/2}, t^{n+1/2}]$, and, in such cases, has the value of 1. As mentioned before, at the interface $[\cdot]_\tau = \pm[\cdot]_\alpha$, where $[\cdot]_\alpha$ denotes spatial jump and the sign depends on the motion of the interface. The operator ∇_h^2 is the standard five point central difference operator and $\boldsymbol{C}_i$, $i = 2, \ldots, 7$, are the spatial correction terms which are only non-zero at the points near the interface. The correction term $\boldsymbol{C}_1$ is the correction term for the discretization of $\partial\boldsymbol{u}/\partial t$ and is only nonzero at a particular grid point which the interface crosses over the time interval $[t^n, t^{n+1}]$.

In our projection method, we need to solve two Helmholtz equations for $\boldsymbol{u}^*$ in (17) and one Poisson equation for ϕ^{n+1} in (21). Since the correction terms in (17) and (21) only affect the right-hand sides of the discrete systems for the Helmholtz and Poisson equations, we can take advantage of the fast solvers from FISHPACK [1] to solve these equations.

4.2. Correction terms

In this section, we will illustrate how to evaluate the correction terms $\boldsymbol{C}_i$, $i = 1, \ldots, 7$ as generated in Section 4.1. We shall define $C\{u\}$ as a correction term for the quantity u. For example, from (9) we can write

$$C\{u_x(x_i)\} = -\frac{1}{2h}\left([u] + h^+[u_x] + \frac{(h^+)^2}{2}[u_{xx}]\right). \tag{25}$$

Then the correction terms $\boldsymbol{C}_1 - C_7$ are evaluated as follows:

$$\boldsymbol{C}_1 = -C\{\boldsymbol{u}_t\} \tag{26}$$

$$\boldsymbol{C}_2 = \frac{3}{2}C\{(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^n\} - \frac{1}{2}C\{(\boldsymbol{u}\cdot\nabla\boldsymbol{u})^{n-1}\} \tag{27}$$

$$\boldsymbol{C}_3 = \frac{1}{2}\left(C\{\nabla^2\boldsymbol{u}^*\} + C\{\nabla^2\boldsymbol{u}^n\}\right) \tag{28}$$

$$\boldsymbol{C}_4 = C\{\nabla p^{n-\frac{1}{2}}\} \tag{29}$$

$$C_5 = \rho\frac{C\{\nabla\cdot\boldsymbol{u}^*\}}{\Delta t} - C\left\{\nabla^2 p^{n+\frac{1}{2}}\right\} + C\left\{\nabla^2 p^{n-\frac{1}{2}}\right\} + \gamma_3[\nabla p]_\tau \tag{30}$$

$$\boldsymbol{C}_6 = -\frac{\Delta t}{\rho}\left(C\left\{\nabla p^{n+\frac{1}{2}}\right\} - C\left\{\nabla p^{n-\frac{1}{2}}\right\}\right) \tag{31}$$

$$C_7 = -\frac{\mu}{2\rho}C\{\nabla\cdot\boldsymbol{u}^*\} \tag{32}$$

All the correction terms are included at least to first order accuracy. As discussed in [20], the overall second order accuracy of the scheme is maintained provided only the singular points are treated with a first order scheme. This can be intuitively understood by observing that when the mesh is refined, the area of the domain represented by these points is reduced.

We note that the correction term $C\{\boldsymbol{u}_t\}$ in (26) is only nonzero at the grid points crossed by the interface between time level n and time level $n+1$. Assume that the interface crosses a grid point (i,j) at time $\tau, t^n \leqslant \tau \leqslant t^{n+1}$, the correction term for $\boldsymbol{u}_t$ at this point is given by

$$C\{\boldsymbol{u}_t\} = -\frac{1}{\Delta t}([\boldsymbol{u}]_\tau + (t^n - \tau)[\boldsymbol{u}_t]_\tau) \tag{33}$$

if $t^n \leqslant \tau \leqslant t^{n+1/2}$, and

$$C\{\boldsymbol{u}_t\} = -\frac{1}{\Delta t}([\boldsymbol{u}]_\tau + (t^{n+1} - \tau)[\boldsymbol{u}_t]_\tau) \tag{34}$$

if $t^{n+1/2} \leqslant \tau \leqslant t^{n+1}$.

Since the velocity is continuous across the interface, we have $[\boldsymbol{u}]_\tau = 0$. Also, by differentiating $[\boldsymbol{u}] = \boldsymbol{0}$ we obtain

$$[\boldsymbol{u}_t] = -[\boldsymbol{u}\cdot\nabla\boldsymbol{u}] = \pm[\boldsymbol{u}_t]_\tau. \tag{35}$$

In (28), (30) and (32), we use the jump conditions for $\boldsymbol{u}^{n+1}$ to approximate the jump conditions for $\boldsymbol{u}^*$ as we expect that $\boldsymbol{u}^*$ is a good approximation for $\boldsymbol{u}^{n+1}$. This is one of the reasons why we have chosen to implement the pressure-increment projection method where $\boldsymbol{u}^*$ is computed to be a good approximation for $\boldsymbol{u}^{n+1}$. To evaluate the correction term $C\{\nabla^2\boldsymbol{u}^*\}$ of (28) at a point (i,j) as depicted in Fig. 2, we need to compute $[\boldsymbol{u}^*_x]$, $[\boldsymbol{u}^*_{xx}]$ at the intersection point α and $[\boldsymbol{u}^*_y]$, $[\boldsymbol{u}^*_{yy}]$ at β using the force strength at time level $n+1$. The correction term $C\{\nabla^2\boldsymbol{u}^*\}$ is calculated as follows:

$$C\{\nabla^2\boldsymbol{u}^*\}_{i,j} = -\frac{[\boldsymbol{u}^*] + h^+[\boldsymbol{u}^*_x]_\alpha + \frac{(h^+)^2}{2}[\boldsymbol{u}^*_{xx}]_\alpha}{h^2} - \frac{[\boldsymbol{u}^*] + k^-[\boldsymbol{u}^*_y]_\beta + \frac{(k^-)^2}{2}[\boldsymbol{u}^*_{yy}]_\beta}{h^2},$$

and $\nabla^2\boldsymbol{u}^*$ is approximated at the point (i,j) as

$$\nabla^2\boldsymbol{u}^*(i,j) = \frac{\boldsymbol{u}^*_{i+1,j} + \boldsymbol{u}^*_{i-1,j} + \boldsymbol{u}^*_{i,j+1} + \boldsymbol{u}^*_{i,j-1} - 4\boldsymbol{u}^*_{i,j}}{h^2} + C\{\nabla^2\boldsymbol{u}^*\}_{i,j} + \mathrm{O}(h).$$

Similarly, we can compute for the other correction terms in (28)–(32).

4.3. Evaluation of singular force at the rigid boundary

Assuming that the singular force $\boldsymbol{f}$ is known at the rigid boundary, the velocity field $\boldsymbol{u}^{n+1}$ at all the grid points can be computed via the projection method as discussed in Section 4.1. In our method, we use a set

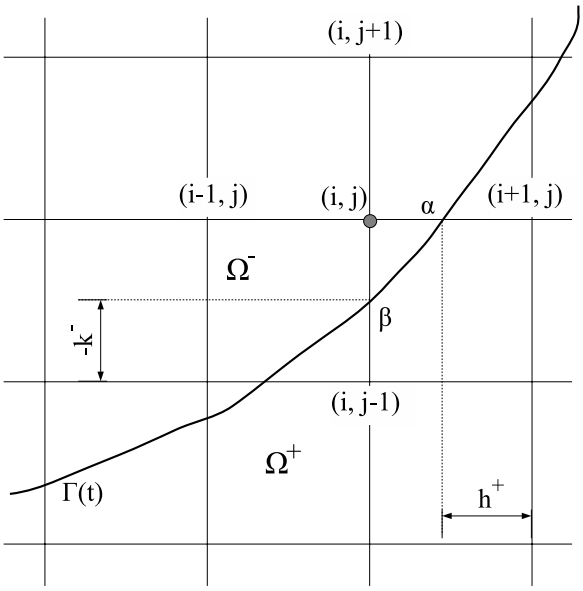


Fig. 2. Interface and mesh geometry near the grid point (i,j).

of control points to represent the rigid boundary. The velocity at the control points, $\boldsymbol{U_k}$, is interpolated from the velocity at the grid points. Thus, we can write

$$\boldsymbol{U_k} = \boldsymbol{U}(\boldsymbol{X}_k) = \mathscr{B}(\boldsymbol{u}^{n+1}), \tag{36}$$

where $\mathscr{B}$ is the bilinear interpolation operator which includes the appropriate correction terms required to guarantee second order accuracy when the derivatives of the velocity are discontinuous. The explicit form of $\boldsymbol{U_k}$ can be found in Appendix A.

In summary, the equations that need to be solved in order to calculate $\boldsymbol{u}^{n+1}$ and $\boldsymbol{U_k}$, can be written symbolically as,

$$\text{Eq. (17)} \;\rightarrow\; \boldsymbol{Hu}^* = \boldsymbol{C} + \boldsymbol{B}_1\boldsymbol{f}\ ,$$
$$\text{Eq. (21)} \;\rightarrow\; \boldsymbol{L\phi}^{n+1} = \boldsymbol{Du}^* + \boldsymbol{B}_2\boldsymbol{f}\ ,$$
$$\text{Eq. (23)} \;\rightarrow\; \boldsymbol{u}^{n+1} = \boldsymbol{u}^* - \boldsymbol{G\phi}^{n+1} + \boldsymbol{B}_3\boldsymbol{f}\ ,$$
$$\text{Eq. (36)} \;\rightarrow\; \boldsymbol{U_k} = \boldsymbol{Mu}^{n+1} + \boldsymbol{B}_4\boldsymbol{f}\ .$$

Eliminating $\boldsymbol{u}^*$, $\boldsymbol{\phi}^{n+1}$ and $\boldsymbol{u}^{n+1}$ from the above equations, we can compute the velocity $\boldsymbol{U_k}$ at the control points as follows:

$$\boldsymbol{U_k} = \boldsymbol{M}(\boldsymbol{H}^{-1}\boldsymbol{C} - \boldsymbol{GL}^{-1}\boldsymbol{DH}^{-1}\boldsymbol{C}) + (\boldsymbol{M}(\boldsymbol{H}^{-1}\boldsymbol{B}_1 - \boldsymbol{GL}^{-1}\boldsymbol{DH}^{-1}\boldsymbol{B}_1 - \boldsymbol{GL}^{-1}\boldsymbol{B}_2 + \boldsymbol{B}_3) + \boldsymbol{B}_4)\boldsymbol{f}\ . \tag{37}$$

For convenience, we can write (37) as

$$\boldsymbol{U_k} = \boldsymbol{U}_k^0 + \boldsymbol{Af}\ , \tag{38}$$

where $\boldsymbol{U}_k^0$ is simply the velocity at the control points obtained by solving Eqs. (17), (21), (23) and (36) with $\boldsymbol{f} = \boldsymbol{0}$, given $\boldsymbol{u}^n$ and $p^{n-1/2}$. $\boldsymbol{A}$ is a $2N_b \times 2N_b$ matrix, where N_b is the number of control points. The vector $\boldsymbol{Af}$ is the velocity at the control points obtained by solving the following equations:

$$\frac{\boldsymbol{u}_f^*}{\Delta t} = \frac{\mu}{2\rho}\nabla^2\boldsymbol{u}_f^*, \quad \boldsymbol{u}_f^*|_{\partial\Omega} = 0, \tag{39}$$

$$\nabla^2\phi_f^{n+1} = \rho\frac{\nabla\cdot\boldsymbol{u}_f^*}{\Delta t}, \quad \boldsymbol{n}\cdot\nabla\phi_f^{n+1}|_{\partial\Omega} = 0, \tag{40}$$

$$\boldsymbol{u}_f^{n+1} = \boldsymbol{u}_f^* - \frac{\Delta t}{\rho}\nabla\phi_f^{n+1}, \tag{41}$$

$$\boldsymbol{A}\boldsymbol{f} = \mathscr{B}(\boldsymbol{u}_f^{n+1}) \tag{42}$$

with $\boldsymbol{f}$ being the singular force at the immersed boundary.

Eq. (38) can be used to determine the singular force if we know the prescribed velocity $\boldsymbol{U}_p$ at the immersed boundary. Thus, the singular force at the control points can be computed by solving

$$\boldsymbol{A}\boldsymbol{f} = \boldsymbol{U}_p - \boldsymbol{U}_k^0. \tag{43}$$

In this way, the singular force is solved to impose exactly the no-slip boundary condition at the interface. In addition, since the singular force is calculated implicitly, the timestep used in our algorithm is usually much larger than that used in other methods with explicit forcing formulations [13,15,31]. Note that the matrix $\boldsymbol{A}$ depends on the location of the interface and the timestep Δt. For static geometry, we will have the same matrix $\boldsymbol{A}$ at every timestep if we use the same Δt throughout. Therefore, the matrix $\boldsymbol{A}$ is computed once and is factorized and stored. In order to compute the coefficients of $\boldsymbol{A}$ we solve Eqs. (39)–(42) for $2N_b$ times, i.e. once for each column. Each time, the force strength $\boldsymbol{f}$ is set to zero except for the entry corresponding to the column we want to calculate, which is set to one. Once the matrix $\boldsymbol{A}$ has been calculated, only the right hand side, $\boldsymbol{U}_p - \boldsymbol{U}_k^0$, needs to be computed at each timestep. The resulting small system of Eq. (43) is then solved at each timestep for the singular force $\boldsymbol{f}$ via back substitution. Finally, we solve Eqs. (17)–(24) to obtain $\boldsymbol{u}^{n+1}$ and $p^{n+1/2}$. It is important to note that the matrix $\boldsymbol{A}$, for a closed immersed boundary, is singular. This happens because the pressure inside the closed boundary is not uniquely determined. We use the singular value decomposition (SVD) method to solve the singular system of Eq. (43).

For moving geometry, the matrix $\boldsymbol{A}$ must be regenerated at every timestep. The computational cost associated with the procedure outlined above would be prohibitive. To avoid generating $\boldsymbol{A}$, we employ GMRES method and solve (43) iteratively. Each iteration of GMRES method requires a matrix–vector product which can be found by solving (39)–(42). In each matrix–vector product, we have to solve two Helmholtz Eq. (39) and a Poisson Eq. (40). Therefore, our algorithm for solving the problems with moving boundary is only effective if the GMRES method takes a few iterations to converge. For a closed immersed boundary, the linear system of Eq. (43) is singular. A version of GMRES method for singular linear system of equations is required. We employed the GMRES method presented in [6] which used the incremental condition estimation (ICE) [3] to monitor the conditioning of the upper Hessenberg matrix.

4.4. Numerical implementation

4.4.1. Rigid boundary

In this section, we describe a basic implementation of our algorithm for the Navier–Stokes equations with immersed rigid boundaries. We describe our approach for the problem of the flow past a circular cylinder. To start our procedure we use a set of control points to represent the rigid boundary and compute the coefficient matrix as mentioned in the previous section. For the cylinder problem, this matrix is singular. We factorize the coefficient matrix using singular value decomposition as,

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}, \tag{44}$$

where $\boldsymbol{u} = [u_1, \ldots, u_N]$ and $\boldsymbol{V} = [v_1, \ldots, v_N]$ are orthogonal matrices and $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_N)$ is a diagonal matrix whose elements are the singular values of the original matrix such that

$$\sigma_1 \geqslant \sigma_2 \geqslant \ldots \geqslant \sigma_N \geqslant 0.$$

Since $\boldsymbol{A}$ is singular it has at least one singular value equals to zero. We store $\boldsymbol{U}$, $\boldsymbol{V}$ and $\boldsymbol{\Sigma}$ for solving the singular force at every timestep. At each timestep, given the velocity field $\boldsymbol{u}^n$ and pressure field $p^{n-1/2}$, our algorithm for finding $\boldsymbol{u}^{n+1}, p^{n+1/2}$ and the singular force to impose the no-slip condition at the rigid boundary can be summarized as follows:

Step 1: Compute the right-hand side of (43) by calculating $\boldsymbol{U}_k^0$.
- Set $\boldsymbol{f} = 0$. Solve (17), (21) and (23) for the velocity at all the grid points.
- Interpolate the velocity at the control points $\boldsymbol{U}_k^0$ as in (36).
- Compute the right-hand side vector $\boldsymbol{b} = \boldsymbol{U}_p - \boldsymbol{U}_k^0$.

Step 2: Compute the singular force by solving (43) using the SVD method.
- If $\boldsymbol{A}$ is nonsingular, then the force $\boldsymbol{f}$ can be written in terms of the SVD as

$$\boldsymbol{f} = \sum_{i=1}^{N} \frac{u_i^T \boldsymbol{b}}{\sigma_i} v_i.$$

- If $\boldsymbol{A}$ is singular and has k singular values due to the presence of k closed rigid boundary loops, then, the force $\boldsymbol{f}$ can be computed as,

$$\boldsymbol{f} = \sum_{i=1}^{N-k} \frac{u_i^T \boldsymbol{b}}{\sigma_i} v_i. \tag{45}$$

Step 3: Compute $\boldsymbol{u}^{n+1}$ and $p^{n+1/2}$ using the projection method.For moving geometry, we still have the same algorithm except that the GMRES solver is applied to solve (43) iteratively at each time step. Thus, we do not need to form the coefficient matrix explicitly.

4.4.2. Motion of flexible boundaries in the presence of rigid boundaries

We now turn our attention to the implementation of the immersed interface method for the incompressible Navier–Stokes equations in general domains involving immersed flexible and rigid boundaries. We consider a generalized force exerted by interface on the fluid of the form

$$\boldsymbol{f}(s,t) = \frac{\partial}{\partial s}(T(s,t)\boldsymbol{\tau}(s,t)) + \sigma \frac{\partial^2 \boldsymbol{X}}{\partial s^2}, \tag{46}$$

where $T(s,t)$ is defined as

$$T(s,t) = T_0\left(\left|\frac{\partial \boldsymbol{X}(s,t)}{\partial s_0}\right| - 1\right) \tag{47}$$

and $\boldsymbol{\tau}(s,t)$ is the unit tangential vector to the interface,

$$\boldsymbol{\tau}(s,t) = \frac{\partial \boldsymbol{X}}{\partial s} \Big/ \left|\frac{\partial \boldsymbol{X}}{\partial s}\right|. \tag{48}$$

Here, $\boldsymbol{X}(s,t)$ is the arc-length parametrization of the interface and s and s_0 are the arc-lengths measured along the current and undeformed configuration of the membrane, respectively. The scalar T_0 is the stiffness constant which describes the elastic property of the flexible boundary. The scalar σ is the surface tension constant. In the case of a flexible boundary governed by surface tension T_0 will be zero, and in the case of an elastic membrane σ will be zero. The location of the flexible boundaries is advanced in time in an implicit manner,

$$\boldsymbol{X}^{n+1} = \boldsymbol{X}^n + \frac{1}{2}\Delta t(\boldsymbol{u}^n(\boldsymbol{X}^n) + \boldsymbol{u}^{n+1}(\boldsymbol{X}^{n+1})). \tag{49}$$

The BFGS method [32] which is a quasi-Newton method is employed to solve the non-linear system of Eq. (49) iteratively to calculate the location of the flexible boundaries. For more details on the immersed interface method for flexible interfaces, see [16,17,19]. In each iteration of the BFGS method, we need to solve the system of Eq. (43) for the singular force at the rigid boundaries to enforce the no-slip boundary conditions. This is necessary because the velocity field and pressure field are updated at every iterations of the BFGS method.

In summary, given the location of the flexible boundaries, $\boldsymbol{X}^n$, the singular force at the rigid boundaries, $\boldsymbol{f}^n$, the velocity field, $\boldsymbol{u}^n$, and the pressure field, $p^{n-1/2}$, the process of computing the new velocity field $\boldsymbol{u}^{n+1}$ that

satisfies the no-slip boundary conditions at the rigid boundaries, pressure field $p^{n+1/2}$ and the location of the flexible boundaries $\boldsymbol{X}^{n+1}$ can be described as follows:

Step 1: Set $k := 0$, set the inverse Jacobian $\boldsymbol{B}_0 = \boldsymbol{I}$ and make an initial guess for $\boldsymbol{X}^{n+1}$, i.e. $\boldsymbol{X}^{(0)}$ as $\boldsymbol{X}^{(0)} = 2\boldsymbol{X}^{\boldsymbol{n}} - \boldsymbol{X}^{\mathbf{n-1}}$.

Step 2:

- Compute the force strength at the flexible boundaries using expression (46). Interpolate the force strength using cubic splines.
- Compute the force strength at the rigid boundaries to enforce the no-slip conditions. That is, calculate the right-hand side vector $\boldsymbol{U}_p - \boldsymbol{U}_k^0$ of (43). Then solve for the small system of Eq. (43) to obtain the singular force at the rigid boundaries.

Step 3:

- Employ the projection method as described in Section 4.1 to update the velocity $\boldsymbol{u}^{n+1}$ and pressure field $p^{n+1/2}$.
- Compute the velocity at the control points, $\boldsymbol{u}^{n+1}(\boldsymbol{X}^{(k)})$, by interpolating from the velocity at the surrounding grid points.

Step 4:

- Evaluate

$$g(\boldsymbol{X}^{(k)}) = \boldsymbol{X}^{(k)} - \boldsymbol{X}^n - \frac{1}{2}\Delta t(\boldsymbol{u}^n(\boldsymbol{X}^n) + \boldsymbol{u}^{n+1}(\boldsymbol{X}^{(k)})).$$

- If $\|g^{(k)}\| < \epsilon$ then $\boldsymbol{X}^{n+1} = \boldsymbol{X}^{(k)}$ and stop the iteration. Otherwise, update $\boldsymbol{X}^{(k+1)}$ and the inverse Jacobian matrix B_{k+1} using BFGS algorithm. Set $k := k + 1$ and go to step 2.

Our implementation prohibits the intersection between two flexible boundaries or between a flexible boundary and rigid boundaries. This is enforced by defining a contact threshold say to be a distance of $1.5h$, where h is the mesh size. If a control point of a flexible boundary lies within a contact threshold of other flexible boundaries or rigid boundaries, we introduce a repulsive force into the total singular force at the flexible boundary. This repulsive force is applied in the outward normal direction to the rigid boundaries or other flexible boundaries. If the flexible membrane represents a particle with surface potential, the repulsive force can be understood as the electrostatic repulsion between two colloidal particles or between a particle and the rigid boundaries. In [2], the velocities of interfacial points that lie within a contact threshold of other membranes or rigid boundaries are adjusted. However, these velocity adjustments may alter the volume of the bodies. A repulsive force is suggested and introduced into the total surface force thereby obviating the need of the velocity adjustment. In our algorithm, the expression for the repulsive force at a control point is

$$|\boldsymbol{f}_R(r)| = \begin{cases} C\left[1 - \left(\frac{r}{1.5h}\right)^n\right], & r \leqslant 1.5h, \\ 0, & \text{otherwise}, \end{cases} \tag{50}$$

where r is a separation distance between a flexible membrane and other flexible membranes or rigid boundaries, C is a positive constant and n is a power index. It can be construed that the values of C and n are functions of the properties of the surface material but this topic is outside the scope of the present study. In our numerical experiments, a typical n is chosen within 2–4 and the constant C is chosen to have the same order of magnitude as the current force at the control point under consideration. In order to avoid a kink which may occur when adding the repulsive force to the singular force at the membrane, we distribute the repulsive force to the nearby control points of the same membrane using a Gaussian normal representation of the discrete delta function. Typically we distribute the repulsive force to five control points including the control point under consideration and its four closest neighbors on the same membrane.

5. Numerical results

In this section, we present the numerical results for several problems involving rigid boundaries and both rigid and flexible boundaries.

5.1. Rotational flow

In this problem, the interface is a circle with radius r = 0.3 embedded in a square domain $[-1,1]\times[-1,1]$. We prescribe the interface to rotate with angular velocity $\omega = 2$. We set $\rho = 1$, $\mu = 0.02$ and consider the solution at $t = 10$. The velocity field is shown in Fig. 3. We carried out a grid refinement analysis, using a referenced grid of 512×512, to determine the order of convergence of the algorithm. The errors in the velocity, $E(\boldsymbol{u})$ and the errors in the pressure, $E(p)$ are measured in both the maximum norm and the second norm. The results in Table 1 show that the velocity is second order accurate and the pressure is nearly second order accurate.

5.2. Flow past a circular cylinder

In this example, we simulate an unsteady flow past a circular cylinder immersed in a rectangular domain $\Omega = [0,3]\times[0,1.5]$. We use this problem as another benchmark test for our algorithm. The cylinder has a diameter $d = 0.1$ and its center is located at (1.6, 0.75). The freestream velocity is set to unity, $U_\infty = 1$ and

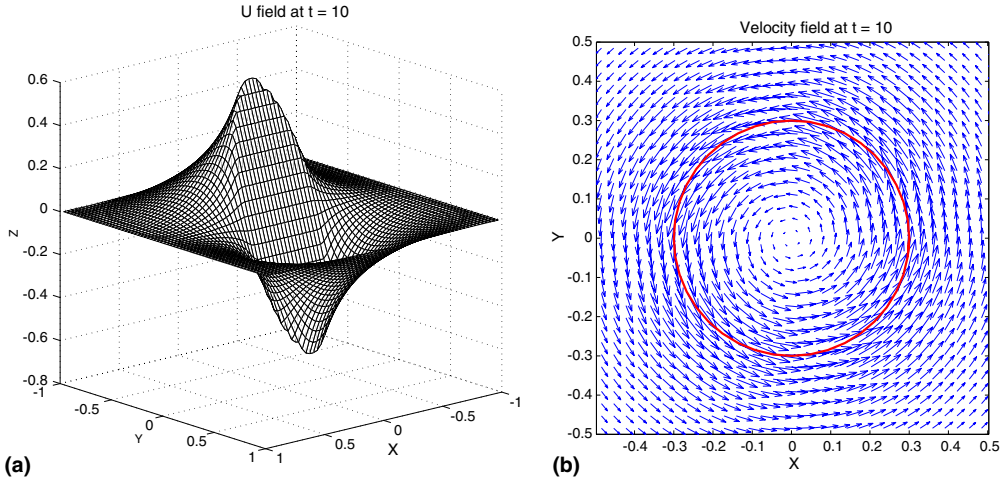


Fig. 3. Velocity field at time $t = 10$ with a 64×64 grid, $\mu = 0.02$, $\Delta t = \Delta x/4$. The immersed boundary rotates with angular velocity $\omega = 2$. (a) Plot of the x component of velocity field. (b) Plot of the velocity field.

Table 1
Grid refinement analysis for the rotational flow problem with $\mu = 0.02$, $\Delta t = \Delta x/4$, at $t = 10$

N	N_b	$\|E(\boldsymbol{u})\|_\infty$	Order	$\|E(\boldsymbol{u})\|_2$	Order
64	40	1.8001×10^{-3}		1.6528×10^{-4}	
128	80	5.5145×10^{-4}	1.71	3.9239×10^{-5}	2.08
256	160	1.2755×10^{-4}	2.11	1.0021×10^{-5}	1.97
		$\|E(p)\|_\infty$		$\|E(p)\|_2$	
64	40	6.6995×10^{-3}		1.6014×10^{-3}	
128	80	1.5951×10^{-3}	2.07	4.7510×10^{-4}	1.75
256	160	5.7996×10^{-4}	1.46	1.5854×10^{-4}	1.58

simulations are carried out at Reynolds number, ($Re = \rho U_\infty d/\mu$) of 20, 40, 80, 100, 200 and 300 on a 512×256 computational mesh. We use 40 points to represent the circular cylinder. At the inflow boundary we specify the velocity corresponding to the freestream velocity, and a homogeneous Neumann boundary condition is applied at the top, bottom and exit boundaries. For all these simulations, we use the free stream velocity as the initial velocity and the initial pressure is set to zero. Then, the force at the cylinder interface is determined by the no-slip condition on the cylinder. The resulting solution inside the cylinder corresponds to zero velocity and a constant pressure which is an arbitrary value. After the first timestep, the flow evolves naturally and satisfies the no-slip boundary condition. Once the velocity field and pressure field have been computed, the drag and lift coefficients and the Strouhal number can be computed from the force at the control points.

The drag coefficient is defined as

$$C_{\mathrm{D}} = \frac{D}{\frac{1}{2}\rho U_\infty^2 d}. \tag{51}$$

The drag can be computed from the force along the cylinder interface as

$$D = -\int_\Gamma f_x \,\mathrm{d}s, \tag{52}$$

where f_x is the x component of the singular force. The lift coefficient is defined as

$$C_{\mathrm{L}} = \frac{L}{\frac{1}{2}\rho U_\infty^2 d}. \tag{53}$$

The lift can be computed from the force along the cylinder interface as

$$L = -\int_\Gamma f_y \,\mathrm{d}s, \tag{54}$$

where f_y is the y component of the singular force. The Strouhal number is defined as

$$St = \frac{fd}{U_\infty}, \tag{55}$$

where f is the vortex shedding frequency; it is one of the key quantities that characterizes the vortex shedding process. This coefficient can be obtained using the Fourier Transform of the periodic variation of the lift coefficient [31]. Finally, the dimensionless time is defined as

$$T = \frac{U_\infty t}{d}. \tag{56}$$

Fig. 4 shows the streamlines for $Re = 20$ and $Re = 40$. For these low Reynolds numbers, the wake formed behind the cylinder gradually attains a steady symmetric state. Once the flow has reached the steady state, the drag coefficients, the length of the recirculation zone and the angle of separation are calculated and are compared with other established results in Table 2. The results obtained by our method are compared to the numerical simulations [7,9,12,29,37] as well as experimental results [8,33]. It is found that our results are in reasonably good agreement with other numerical simulations and experimental results. For $Re = 20$ our drag coefficient is very closed to other numerical results but it is about 8% lower than the experimental measurement of Tritton [33]. For $Re = 40$ our drag coefficient is about 5% higher than the experimentally determined value [33]. Fig. 5 shows the plots of the pressure field for $Re = 20$ and $Re = 40$. The pressure patterns are symmetric about the streamwise axis.

Between $Re = 40$ and $Re = 50$ we expect to see a transition to instability. Fig. 6 shows that our algorithm is able to detect the onset of an instability in the flow at $Re = 50$. It has been reported that the wake behind the cylinder first becomes unstable at a critical Reynolds number of about $Re = 46 \pm 1$ [37]. Above this Reynolds number the cylinder wake instability appears and grows in time and eventually leads to Karman vortex shedding. This behavior is shown in the present numerical simulations for $Re = 80$, 100, 200 and 300. Note that in all these simulations we do not need to artificially perturb the flow field to initiate the unsteady behavior. Fig. 7 shows the pressure fields at $Re = 100$, 200 and 300. The instabilities and vortex shedding can be

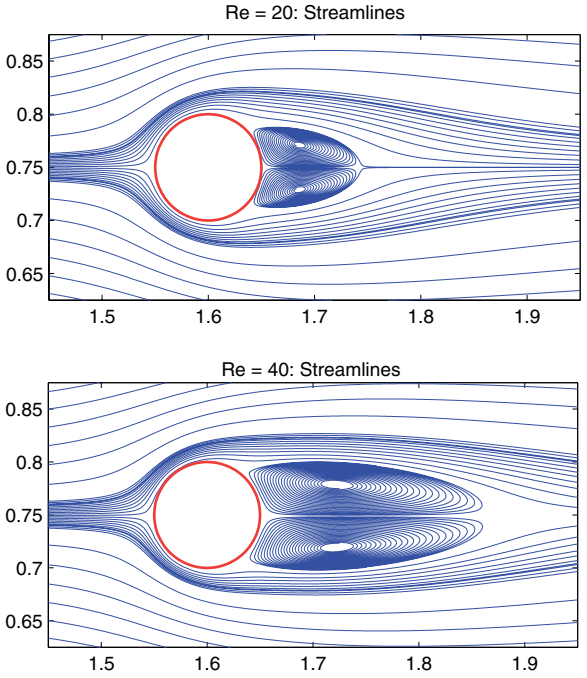


Fig. 4. Streamlines for $Re = 20$ and $Re = 40$.

Table 2
Length of the recirculation zone (L/d), angle of separation (θ) and drag coefficient (C_D) for $Re = 20$ and $Re = 40$

	$Re = 20$			$Re = 40$		
	L/d	θ	C_D	L/d	θ	C_D
Tritton [33]	–	–	2.22	–	–	1.48
Coutanceau and Bouard [8]	0.73	42.3°	–	1.89	52.8°	–
Fornberg [12]	0.91	–	2.00	2.24	–	1.50
Dennis and Chang [9]	0.94	43.7°	2.05	2.35	53.8°	1.52
Calhoun [7]	0.91	45.5°	2.19	2.18	54.2°	1.62
Russell and Wang [29]	0.94	43.3°	2.13	2.29	53.1°	1.60
Ye et al. [37]	0.92	–	2.03	2.27	–	1.52
Present	0.93	43.9°	2.05	2.22	53.6°	1.56

visualized from this figure. In Tables 3 and 4, the drag and lift coefficients at $Re = 100$ and $Re = 200$ are compared to other numerical simulations. For $Re = 100$, the mean drag obtained by our algorithm is slightly greater than that computed by other researchers [4,7,24]. Our drag coefficient differs from that reported by 1–3%. For $Re = 200$, our drag coefficient lies within the range of results reported in [4,7,24,29]. Our value is about 15% higher than that in Calhoun [7] and 4% lower than the value obtained by Braza et al. [4]. In Table 4 it can be seen that the lift coefficient calculated by our method for $Re = 100$ is well within the range of the values obtained by other researchers. However our lift coefficient for $Re = 200$ is lower than their values. Figs. 8 and 9 show the variations in time for the drag and lift coefficients, respectively. These figures depict the development of the vortex shedding to a periodic state with time at $Re = 100$ and $Re = 200$. The vortex shedding Strouhal number is computed for $Re = 80$, 100, 200 and 300 and is compared with other established results in Table 5. Our computed Strouhal number obtained at $Re = 80$ comes out to be 0.15 which compares very well with the values obtained from experiment [36] and from numerical simulation [37]. At $Re = 100$ and $Re = 200$, our Strouhal numbers are in good agreement with those given in [7,24,29] and differ from the exper-

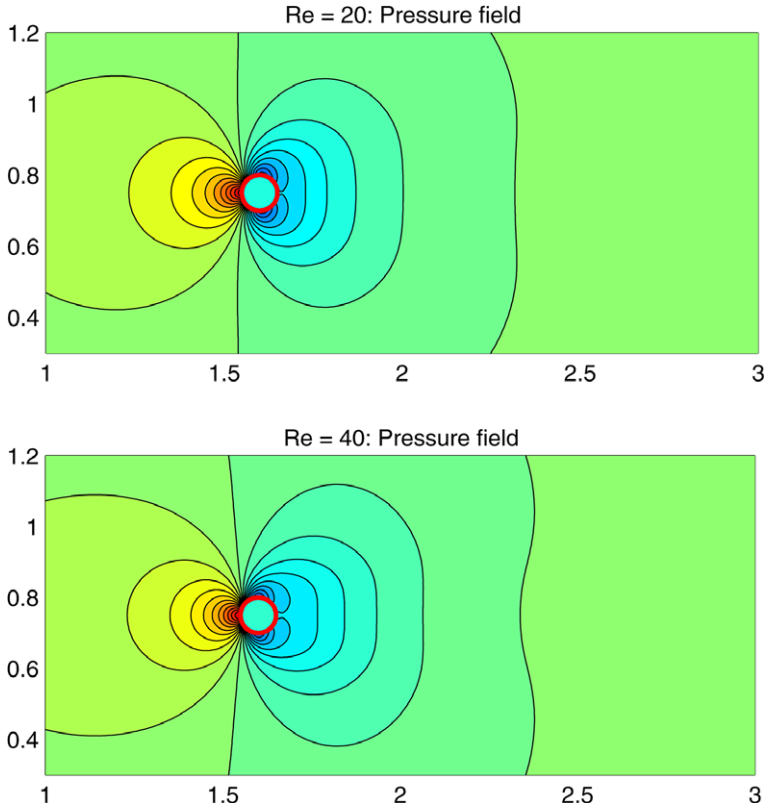


Fig. 5. Pressure fields for $Re = 20$ and $Re = 40$.

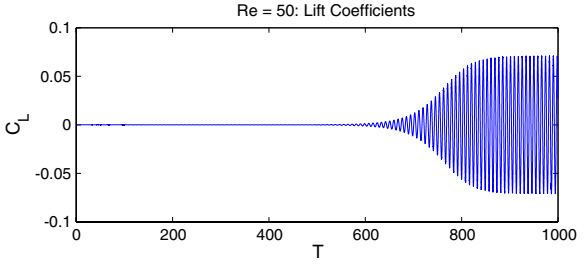


Fig. 6. Lift coefficients at $Re = 50$.

imental results [36] by 1.8% and 1%, respectively. At $Re = 300$, our computed Strouhal number compares very well with the value obtained from experiment [36].

5.3. Flow past several cylinders

In this example, we consider an unsteady flow past several cylinders immersed in a rectangular domain $\Omega = [0,3] \times [0, 1.5]$. This example shows the ability of our algorithm to handle multiple rigid boundaries. The simulation has been performed for three cylinders immersed in the flow at $Re = 100$. All the cylinders have the same diameter of 0.1 and their centers are at $(1.0, 0.75)$, $(1.2, 0.65)$ and $(1.3, 0.85)$. We use 20 control points to represent each of the circular cylinders. The computational grid is 512×256 and the same boundary conditions as those for the flow past a single cylinder problem are applied. Figs. 10 and 11 show the stream-

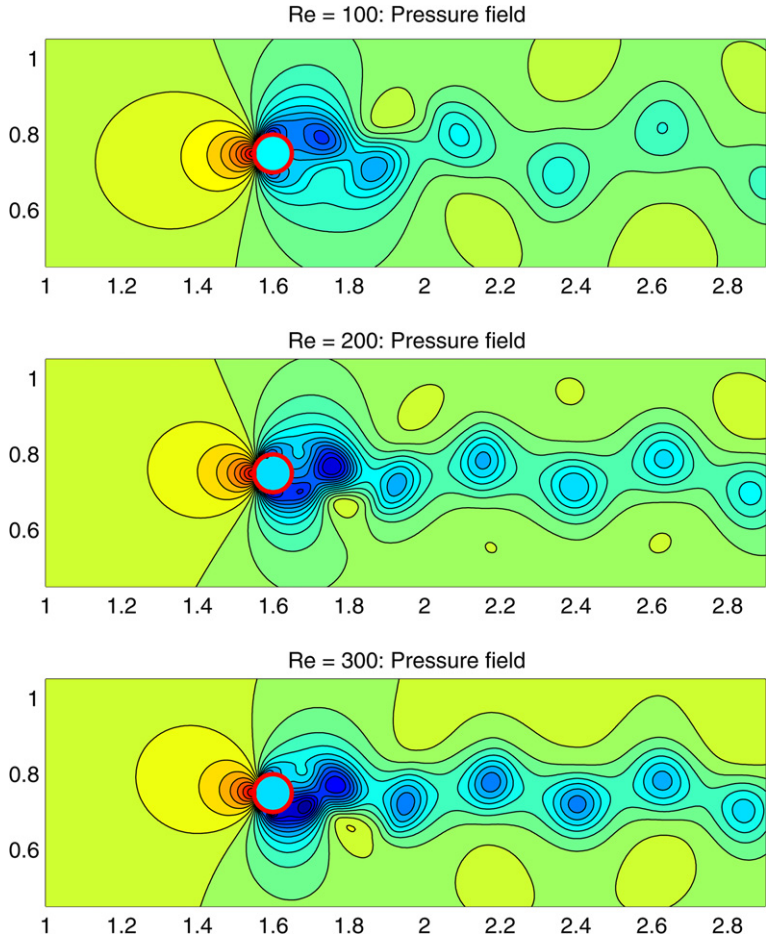


Fig. 7. Pressure fields for $Re = 100$, $Re = 200$ and $Re = 300$.

Table 3
Drag coefficients for $Re = 100$ and $Re = 200$

C_D	$Re = 100$	$Re = 200$
Braza et al. [4]	1.36 ± 0.015	1.40 ± 0.050
Liu et al. [24]	1.35 ± 0.012	1.31 ± 0.049
Calhoun [7]	1.33 ± 0.014	1.17 ± 0.058
Russell et al. [29]	1.38 ± 0.007	1.29 ± 0.022
Present	1.37 ± 0.009	1.34 ± 0.030

Table 4
Lift coefficients for $Re = 100$ and $Re = 200$

C_L	$Re = 100$	$Re = 200$
Braza et al. [4]	± 0.250	± 0.75
Liu et al. [24]	± 0.339	± 0.69
Calhoun [7]	± 0.298	± 0.67
Russell et al. [29]	± 0.300	± 0.50
Present	± 0.323	± 0.43

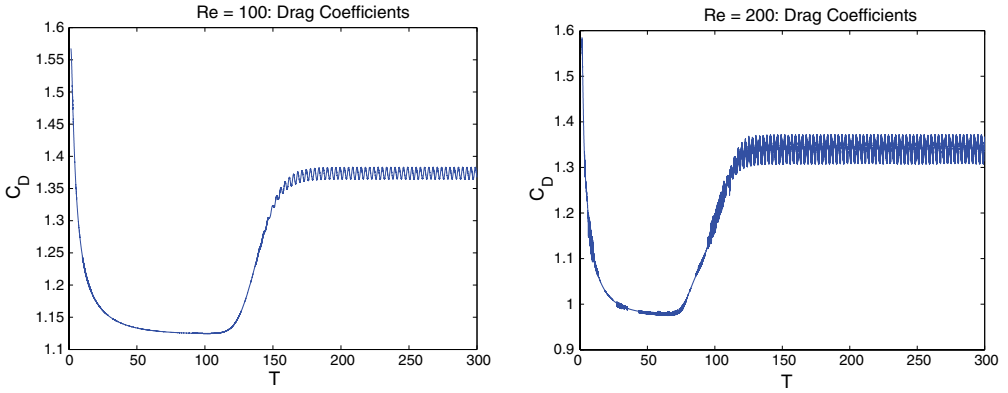


Fig. 8. Drag coefficients for $Re = 100$ and $Re = 200$.

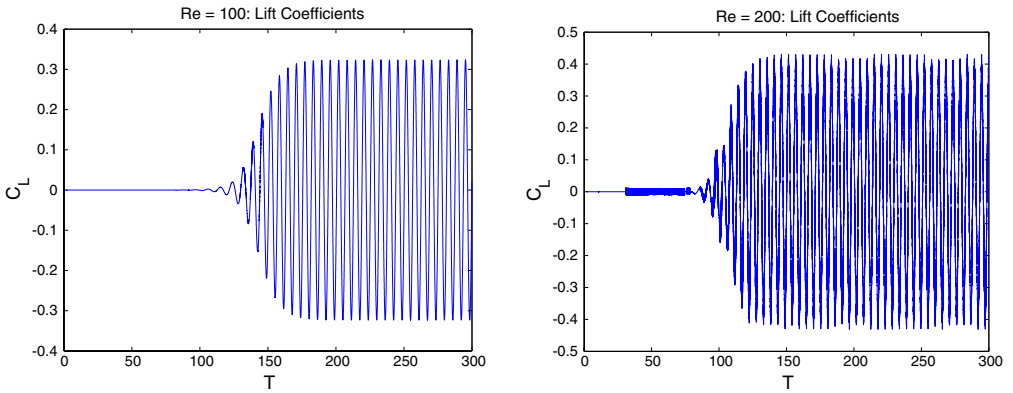


Fig. 9. Lift coefficients for $Re = 100$ and $Re = 200$.

Table 5
Strouhal numbers for $Re = 80$, 100, 200 and 300

St	$Re = 80$	$Re = 100$	$Re = 200$	$Re = 300$
Ye et al. [37]	0.15	–	–	0.210
Williamson [36]	0.15	0.163	0.185	0.203
Liu et al. [24]	–	0.164	0.192	–
Calhoun [7]	–	0.175	0.202	–
Russell et al. [29]	–	0.169	0.195	–
Present	0.15	0.160	0.187	0.200

lines and pressure contours for $Re = 100$ at different time levels. The vortex shedding is not symmetric since the cylinders are not placed symmetrically.

5.4. Flow past a moving circular cylinder

In this example, we simulate the flow past a moving cylinder which moves to the left at a velocity of $U_\infty = -1$. The computational domain is $[0,6]\times[-1.5,1.5]$. The cylinder has a radius $r = 0.1$ and its center is initially located at (5.5, 0.0). At the left boundary we set the velocity to zero, and a homogeneous Neumann boundary condition is applied at the top, bottom and right boundaries. In the frame of reference that is

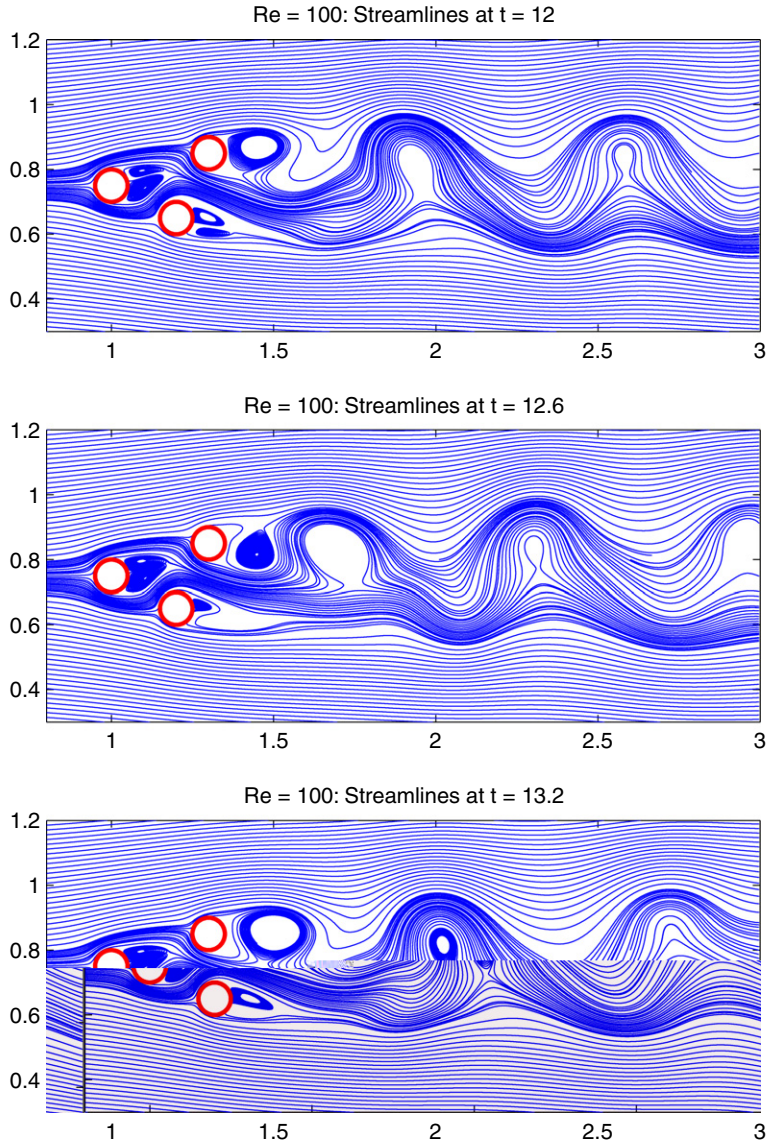


Fig. 10. Flow past three cylinders. Streamline plots for $Re = 100$ at different times.

attached to the moving cylinder, these boundary conditions are the same as those used for the stationary circular cylinder problem. The simulation has been performed for $Re = 40$.

In this example, to solve for the singular force at the moving boundary, we do not generate a system of equations explicitly. Instead, we solve for the force at the boundary iteratively via GMRES algorithm. Since the system of equations is singular, the convergence rate of the GMRES algorithm is relatively slower than for non-singular case. However, we can use the incremental condition estimation (ICE) [3] to monitor the conditioning of the upper Hessenberg matrix and stop the iteration when the conditioner number increases rapidly or when the residual does not change much. Numerical experiments show that the residual is reasonably small and decreases very little after 2–5 iterations. Hence, we can typically stop the GMRES iterative process after 2–5 iterations to reduce numerical effort.

Fig. 12 shows the streamlines plot for $Re = 40$ in the frame of reference attached to the moving cylinder when the wake behind the cylinder appears to be fully developed. Fig. 13 shows the streamlines plot at the

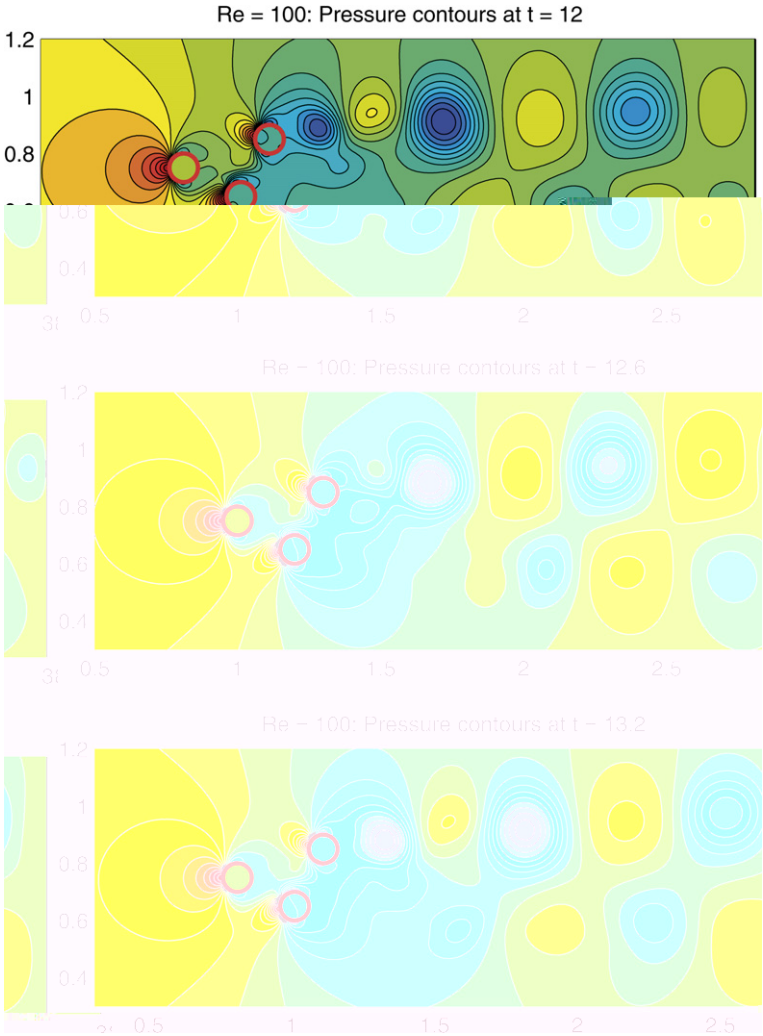


Fig. 11. Flow past three cylinders. Pressure contours for $Re = 100$ at different times.

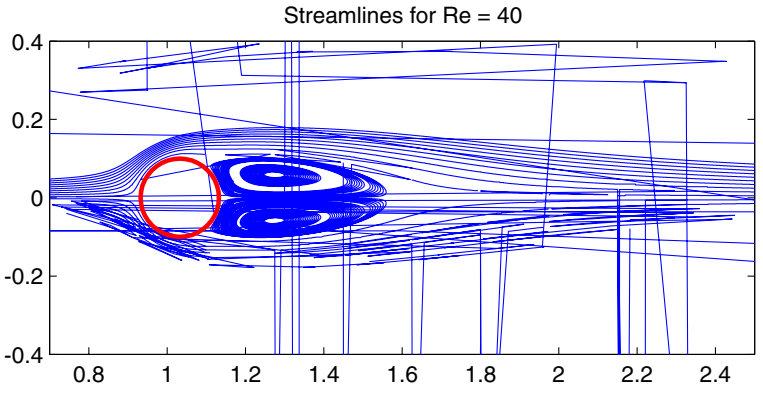
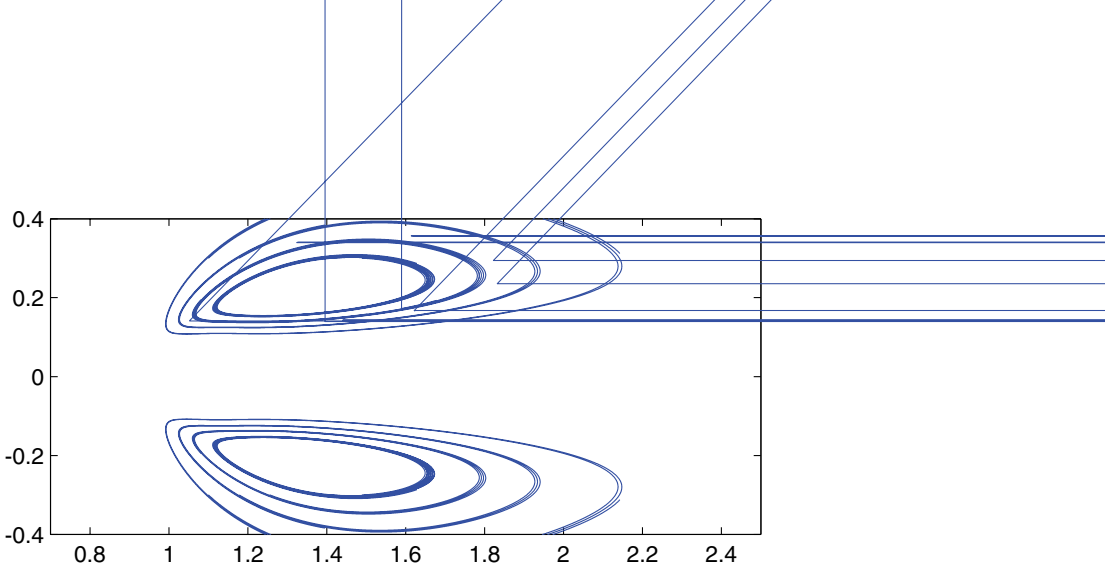


Fig. 12. Streamlines for moving cylinder at $Re = 40$ in the frame of reference attached to the moving cylinder when the wake behind the cylinder is fully developed.

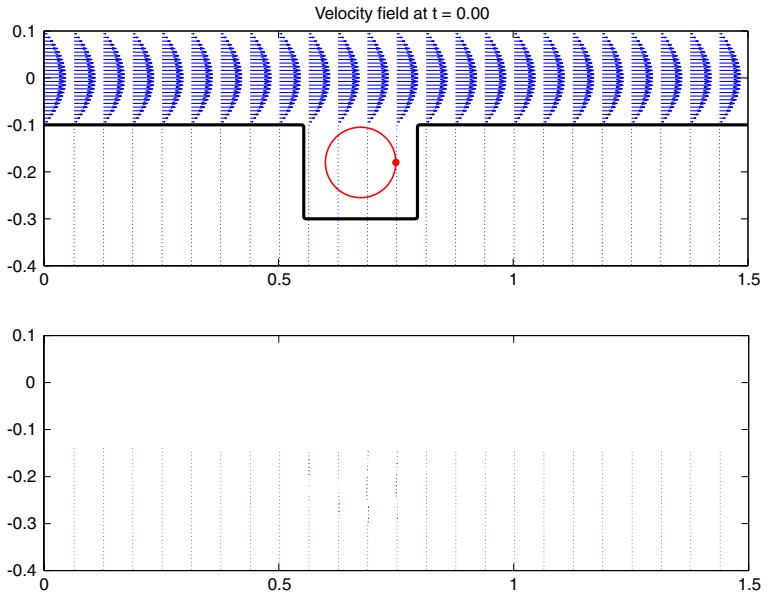


same time level. Table 6 shows the results of the drag coefficient and the length of the recirculation zone at $Re = 40$. These results are compared to those obtained for the stationary cylinder. We can see that the length of the recirculation zone is in reasonable agreement with that obtained for the stationary cylinder. The drag coefficient is about 6.5% higher than that obtained for the stationary cylinder.

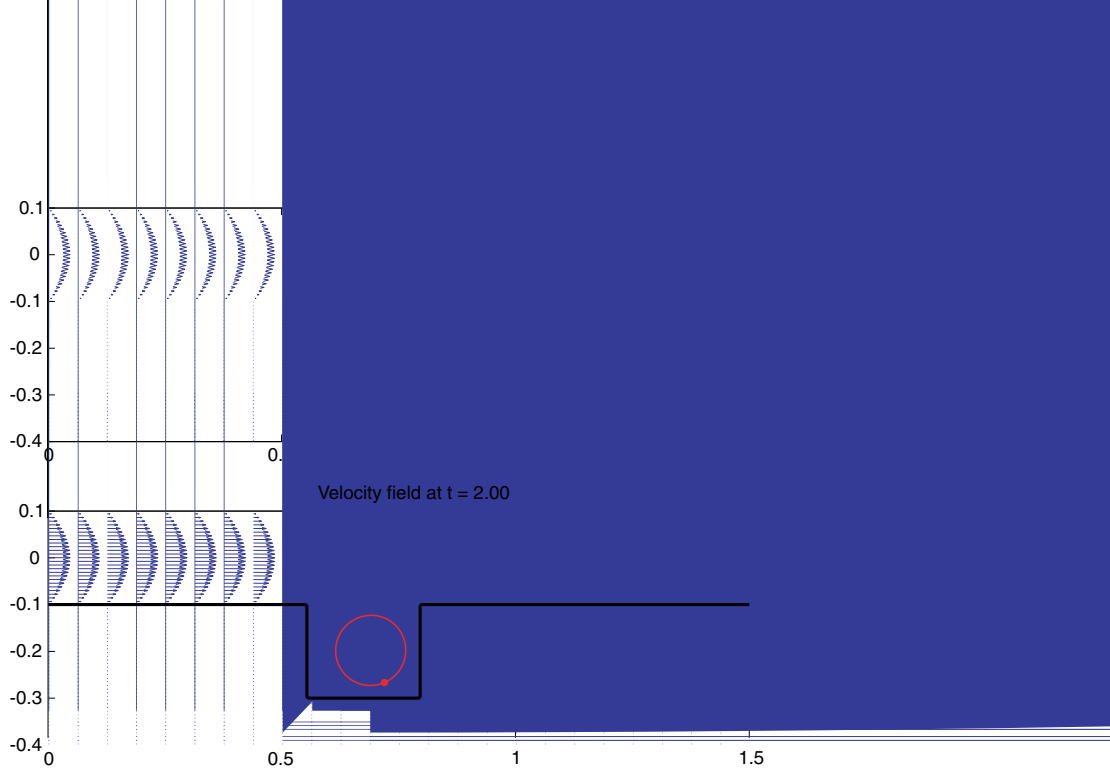
5.5. Motion of elastic membranes in irregular domains

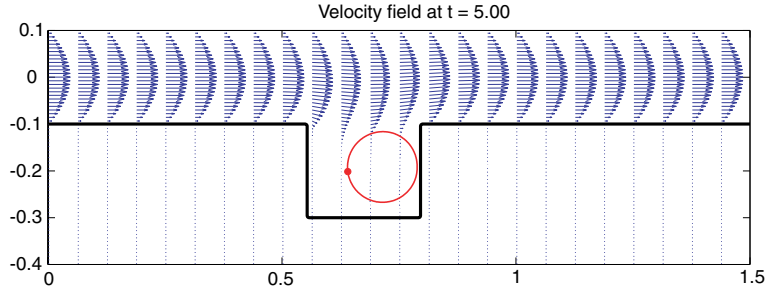
5.5.1. Grooved channel flow with an immersed elastic membrane

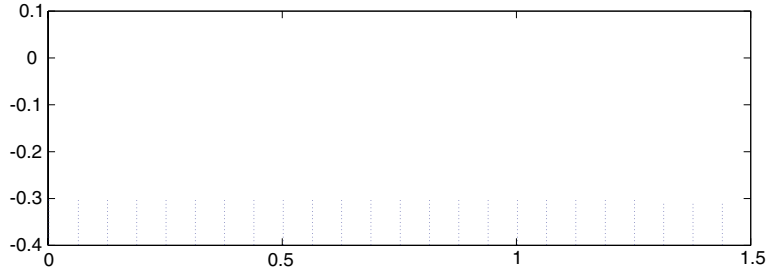
This example considers the Poiseuille flow between two walls, one of which has a groove perpendicular to the streamwise direction. An elastic membrane is immersed in the fluid inside the groove. Depending on the flow, the elastic membrane rotates inside the groove or the flow can move it out of the groove depending on some parameters such as the initial location of the elastic membrane, the flow rate, the size of the groove and the stiffness of the membrane. In the numerical simulation, the gap between the walls is 0.2, the depth and the width of the groove are denoted by D and W, respectively. The velocity profile at the inflow boundary is parabolic with maximum velocity $U_{\max}$, the density is set equal to one, and a viscosity is $\mu = 0.02$. Fig. 14 illustrates the geometry of the grooved channel and the initial position of the membrane inside the groove. A homogeneous Neumann boundary condition for velocity is applied at the right boundary. The velocity is set to zero at the top and bottom boundaries. The no-slip boundary condition at the immersed rigid boundary is enforced by imposing an appropriate singular force at the rigid boundary.


Velocity field at t = 0.00
0.1
0
-0.1
-0.2
-0.3
-0.4
0
0.5
1
1.5

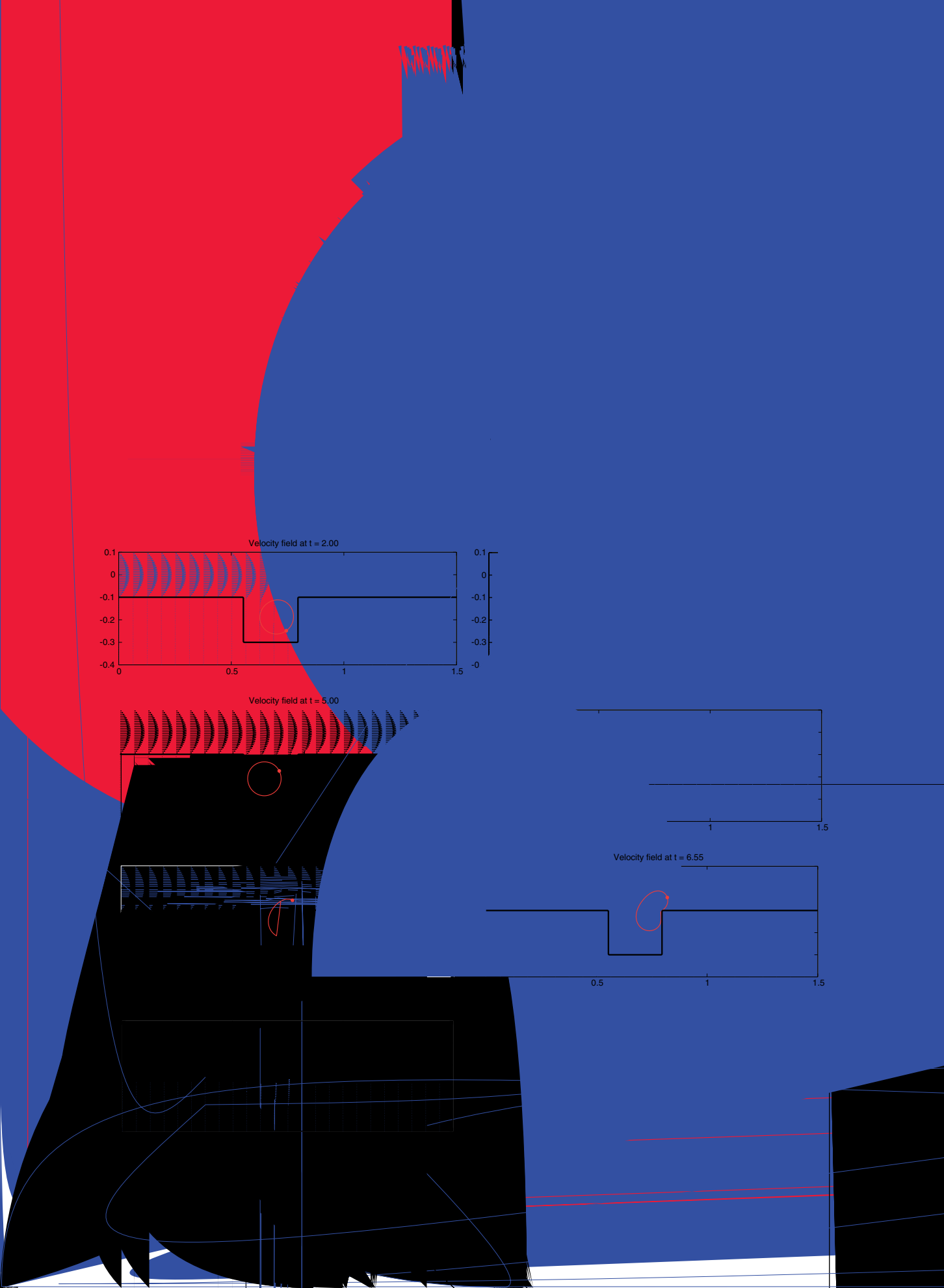
0.1
0
-0.1
-0.2
-0.3
-0.4
0
0.5
1
1.5


0.1
0
-0.1
-0.2
-0.3
-0.4
0
0.5
1
1.5
Velocity field at t = 2.00


Velocity field at t = 5.00
0.1
0
-0.1
-0.2
-0.3
-0.4
0
0.5
1
1.5


0.1
0
-0.1
-0.2
-0.3
-0.4
0
0.5
1
1.5

0
0.5

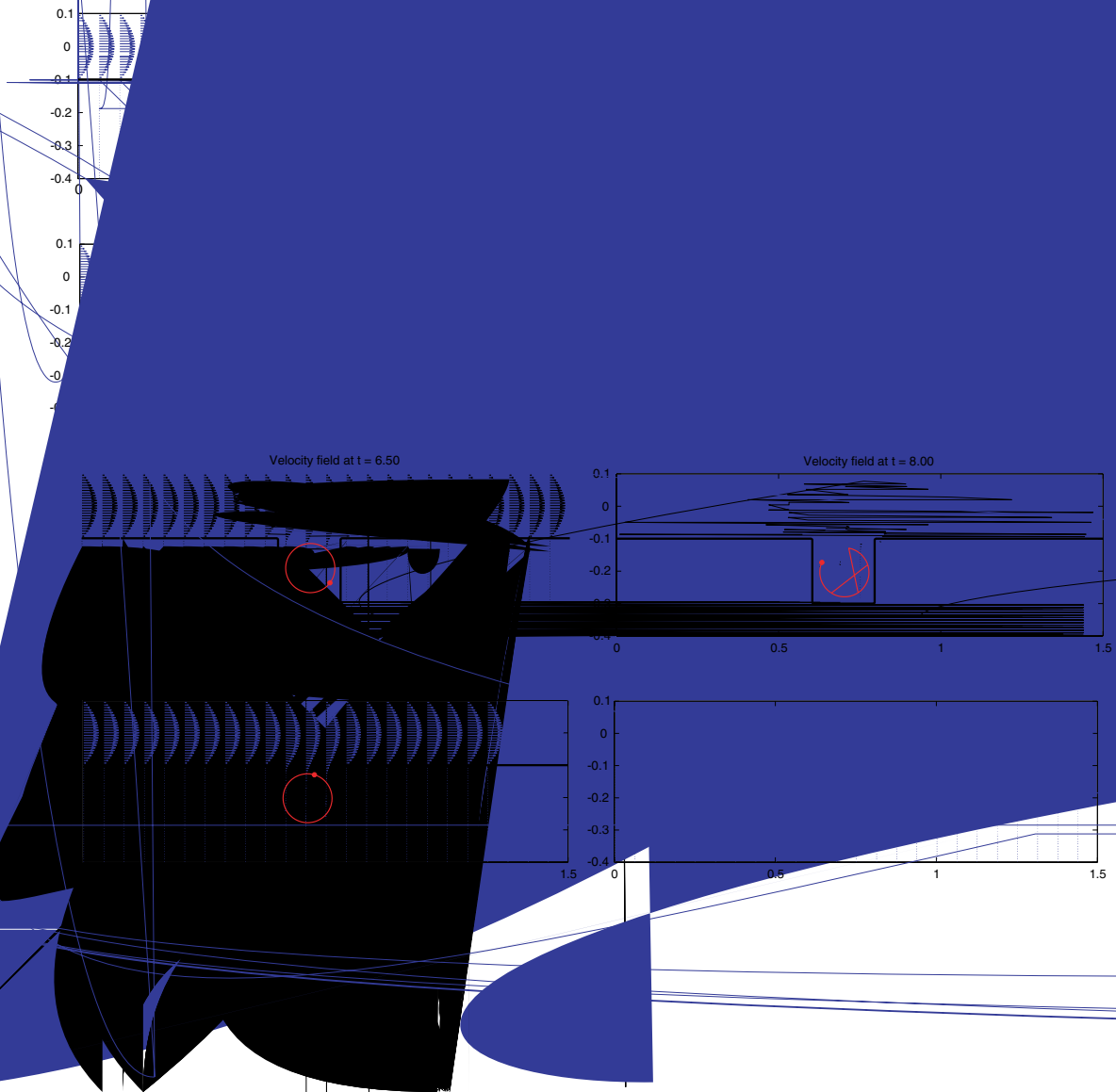

Velocity field at t = 2.00
0.1
0
-0.1
-0.2
-0.3
-0.4
0
0.5
1
1.5
Velocity field at t = 5.00
Velocity field at t = 6.55
0.5
1
1.5

membrane. In these simula
1.5 h and 3 h for the first
this paper were performe

Next, we keep the loca
the maximum far-field ve
out of the groove. Fig. 1
of the high flow rate a
membrane as it tries t

Under the same flo
membrane inside the
inside the smaller gr

We note that whi
these are not observ
sistent modelling c
ensures the exact
appears to be the



5.5.2. Flow in a constriction with immersed elastic membranes

This problem considers the motion of one or more membranes in a domain with a constriction. Fig. 19 illustrates the geometry of the constriction and the initial position of a single membrane in front of the constriction. In all the simulations presented in this example, a computational domain of $[0, 1.5] \times [-0.25, 0.25]$, a 384×128 grid, a fluid density of unity, a fluid viscosity of 0.02 and a surface tension constant of $\sigma = 0$ have been used. A parabolic velocity profile with $U_{\max} = 1$ is specified for the velocity at the inflow boundary. A homogeneous Neumann boundary condition for velocity is applied at the right boundary. The velocity is set to zero at the top and bottom boundaries. The no-slip boundary condition at the immersed rigid boundaries is enforced by imposing appropriate singular forces at the rigid boundaries.

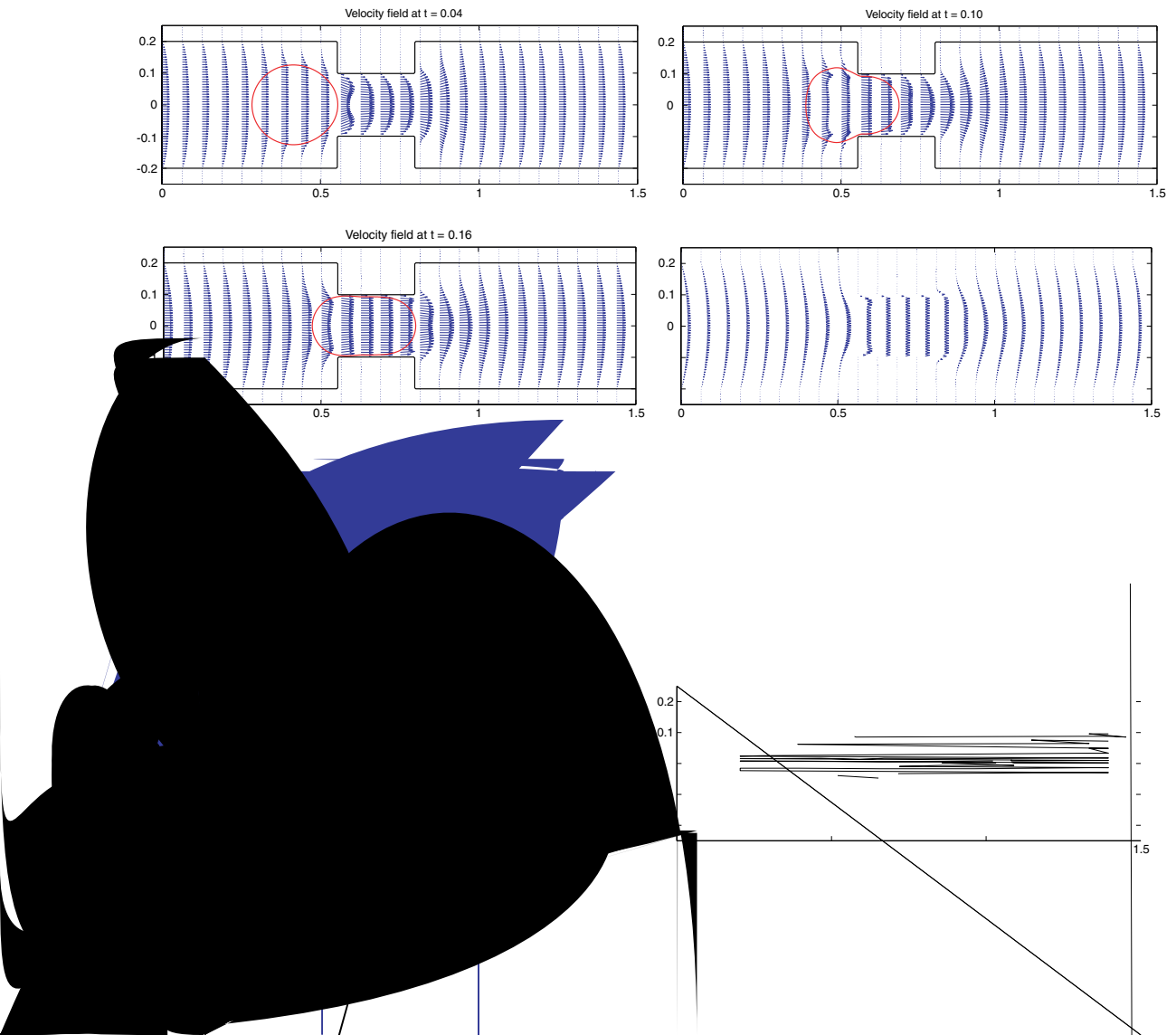


Fig. 218 A single elastic membrane with stiffness constant $T = 4$ squeezes through a constriction with aspect ratio $\frac{1}{2}$.

and unsteady regimes. Moving rigid boundaries are also considered to show the flexibility of our algorithm. Finally, simulations were also performed for problems involving the motion of a single membrane in a grooved channel and single and multiple membranes moving through a constriction.

The presented algorithm can be extended to three dimensions in a rather straightforward manner. In this case the interface would be represented by a surface triangulation and the control point would be the nodes of the triangulation. An issue that needs to be addressed further is the computation of the interaction forces when two membranes approach each other or when a membrane comes close to the rigid boundary. Since one of the main motivations of the current work is the motion of deformable particles in biological flows, the colloidal interaction force between two particles or between a particle with rigid boundaries is likely a combination of Van de Waals attractive force, electrostatic repulsive force and short-ranged Born repulsive force [28].

Appendix A. Modified bilinear interpolation

In this appendix, we derive a bilinear interpolation formula to compute the velocity at a control point. The velocity at the control points, $\boldsymbol{U_k}$, is interpolated from the velocity at the nearby Cartesian grid points. Thus, we can write

$$\boldsymbol{U_k} = \boldsymbol{U}(\boldsymbol{X}_k) = \mathscr{B}(\boldsymbol{u}), \tag{57}$$

where $\mathscr{B}$ is the bilinear interpolation operator which includes the appropriate correction terms which are required to guarantee second order accuracy when the derivatives of the velocity are discontinuous. In Fig. 24, the velocity at the control point $\boldsymbol{X}_k$ is interpolated from the velocity at the four neighboring grid points as follows:

$$\boldsymbol{U}_k = (1-\xi)(1-\eta)\boldsymbol{u}_1 + \boldsymbol{C}_1 + \xi(1-\eta)\boldsymbol{u}_2 + \boldsymbol{C}_2 + \xi\eta\boldsymbol{u}_3 + \boldsymbol{C}_3 + (1-\xi)\eta\boldsymbol{u}_4 + \boldsymbol{C}_4 \tag{58}$$

where $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_4$ are correction terms, $\xi = \frac{X-x_1}{h}$, $\eta = \frac{Y-y_1}{h}$ and h is the grid size. Jump conditions $[\boldsymbol{u}_x]$ and $[\boldsymbol{u}_y]$ are required at the control point to compute the correction terms. The correction terms can be derived using Taylor series expansion and have the following forms:

$$\boldsymbol{C}_1 = \begin{cases} h(1-\xi)(1-\eta)(\xi[\boldsymbol{u}_x] + \eta[\boldsymbol{u}_y]), & \boldsymbol{x}_1 \in \Omega^+, \\ 0, & \boldsymbol{x}_1 \in \Omega^-, \end{cases} \tag{59}$$

$$\boldsymbol{C}_2 = \begin{cases} -h\xi(1-\eta)((1-\xi)[\boldsymbol{u}_x] - \eta[\boldsymbol{u}_y]), & \boldsymbol{x}_2 \in \Omega^+, \\ 0, & \boldsymbol{x}_2 \in \Omega^-, \end{cases} \tag{60}$$

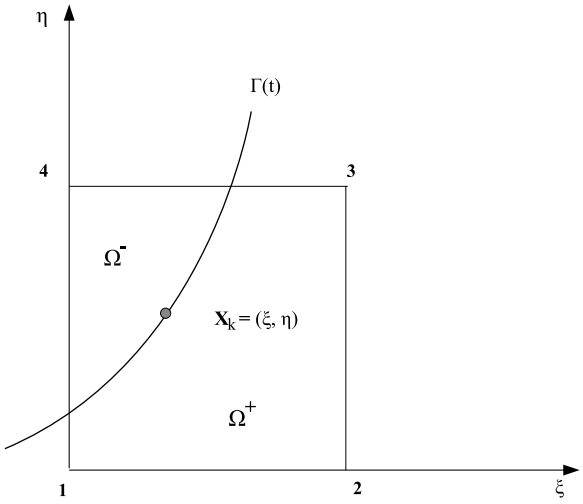


Fig. 24. Velocity at a control point is interpolated from the velocity at the four neighboring grid points using a modified bilinear interpolation.

$$\boldsymbol{C}_3 = \begin{cases} -h\xi\eta((1-\xi)[\boldsymbol{u}_x] + (1-\eta)[\boldsymbol{u}_y]), & \boldsymbol{x}_3 \in \Omega^+, \\ 0, & \boldsymbol{x}_3 \in \Omega^-, \end{cases} \tag{61}$$

$$\boldsymbol{C}_4 = \begin{cases} h(1-\xi)\eta(\xi[\boldsymbol{u}_x] - (1-\eta)[\boldsymbol{u}_y]), & \boldsymbol{x}_4 \in \Omega^+, \\ 0, & \boldsymbol{x}_4 \in \Omega^-. \end{cases} \tag{62}$$

References

[1] J. Adams, P. Swarztrauber, R. Sweet, FISHPACK: Efficient FORTRAN subprograms for the solution of separable eliptic partial differential equations, 1999. Available on the web at http://www.scd.ucar.edu/css/software/fishpack/..

[2] G. Agresar, J.J. Linderman, G. Tryggvason, K.G. Powell, An adaptive, Cartesian, front-tracking method for the motion, deformation and adhesion of circulating cells, J. Comput. Phys. 143 (1998) 346–380.

[3] C.H. Bischof, Incremental condition estimation, SIAM J. Matrix Anal. Appl. 11 (1990) 312–322.

[4] M. Braza, P. Chassaing, H. Ha Minh, Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder, J. Fluid. Mech. 165 (1986) 79–130.

[5] D.L. Brown, R. Cortez, M.L. Minion, Accurate projection methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 168 (2001) 464–499.

[6] P.N. Brown, H.F. Walker, GMRES on (nearly) singular systems, SIAM J. Matrix Anal. Appl. 18 (1) (1997) 37–51.

[7] D. Calhoun, A Cartesian grid method for solving the two-dimensional streamfunction-vorticity equations in irregular regions, J. Comput. Phys. 176 (2002) 231–275.

[8] M. Coutanceau, R. Bouard, Experimental determination of the main features of the viscous flow in the wake of a circular cylinder in uniform translation. Part 1. Steady flow, J. Fluid. Mech. 79 (2) (1977) 231–256.

[9] S.C.R. Dennis, G. Chang, Numerical solutions for steady flow past a circular cylinder at Reynolds number up to 100, J. Fluid. Mech. 42 (3) (1970) 471–489.

[10] R. Dillon, L.J. Fauci, D. Graver, A microscale model of bacterial swimming, chemotaxis and substrate transport, J. Theor. Biol. 177 (1995) 325–340.

[11] A.L. Fogelson, Continuum models of platelet aggregation: Formulation and mechanical properties, SIAM J. Applied Math. 52 (1992) 1089–1110.

[12] B. Fornberg, A numerical study of steady viscous flow past a circular cylinder, J. Fluid. Mech. 98 (4) (1980) 819–855.

[13] D. Goldstein, R. Handler, L. Sirovich, Modeling a no-slip flow with an external force field, J. Comput. Phys. 105 (1993) 354–366.

[14] J. Kim, P. Moin, Application of a fractional step method to incompressible Navier–Stokes equations, J. Comput. Phys. 59 (1985) 308–323.

[15] M.C. Lai, C.S. Peskin, An immersed boundary method with formal second order accuracy and reduced numerical viscosity, J. Comput. Phys. 160 (2000) 707–719.

[16] D.V. Le, An immersed interface method for solving viscous incompressible flows involving rigid and flexible boundaries, PhD thesis, Singapore-MIT Alliance, June 2005..

[17] D.V. Le, B.C. Khoo, J. Peraire, An immersed interface method for the incompressible Navier–Stokes equations, Presented at the Singapore-MIT Alliance (SMA) Symposium, Singapore 2004..

[18] D.V. Le, B.C. Khoo, J. Peraire, An immersed interface method for the incompressible Navier–Stokes equations in irregular domains, in: K.J. Bathe (Ed.), Proceedings of the Third M.I.T. Conference on Computational Fluid and Solid Mechanics, Elsevier Science, 2005, pp. 710–716.

[19] L. Lee. An immersed interface method for incompressible Navier–Stokes equations, SIAM J. Sci. Comput. 25 (3) (2003) 832–856.

[20] R.J. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 31 (1994) 1019–1044.

[21] R.J. LeVeque, Z. Li, Immersed interface method for Stokes flow with elastic boundaries or surface tension, SIAM J. Sci. Comput. 18 (3) (1997) 709–735.

[22] Z. Li, M.C. Lai, The immersed interface method for the Navier–Stokes equations with singular forces, J. Comput. Phys. 171 (2001) 822–842.

[23] Z. Li, C. Wang, A fast finite difference method for solving Navier–Stokes equations on irregular domains, Comm. Math. Sci. 1 (1) (2003) 180–196.

[24] C. Liu, X. Sheng, C.H. Sung, Preconditioned multigrid methods for unsteady incompressible flows, J. Comput. Phys. 139 (1998) 35–57.

[25] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (1977) 220–252.

[26] C.S. Peskin, The immersed boundary method, Acta Numerica 11 (2) (2002) 479–517.

[27] S. Popinet, S. Zaleski, A front-tracking algorithm for accurate representation of surface tension, Int. J. Numer. Meth. Fluids 30 (1999) 775–793.

[28] V. Ramachandran, R. Venkaresan, G. Tryggvason, H.S. Fogler, Low Reynolds number interactions between colloidal particles near the entrance to a cylindrical pore, J. Colloid Interface Sci 229 (2000) 311–322.

[29] D. Russell, Z.J. Wang, A Cartesian grid method for modeling multiple moving objects in 2D incompressible viscous flow, J. Comput. Phys. 191 (2003) 177–205.

[30] C.W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock capturing scheme, II, J. Comput. Phys. 83 (1989) 32–78.
[31] A.L.F. Lima, E. Silva, A. Silveira-Neto, J.J. R Damasceno, Numerical simulation of two-dimensional flows over a circular cylinder using the immersed boundary method, J. Comput. Phys. 189 (2003) 351–370.
[32] J. Stoer, R. Bulirsch, Introduction to Numerical Analysis, 3rd ed., Springer-Verlag, 2002.
[33] D.J. Tritton, Experiments on the flow past a circular cylinder at low Reynolds numbers, J. Fluid. Mech. 6 (4) (1959) 547–567.
[34] N.T. Wang, A.L. Fogelson, Computational methods for continuum models of platelet aggregation, J. Comput. Phys. 151 (1999) 649–675.
[35] A. Wiegmann, K.P. Bube, The explicit-jump immersed interface method: Finite difference methods for PDEs with piecewise smooth solutions, SIAM J. Numer. Anal. 37 (3) (2000) 827–862.
[36] C.H.K. Williamson, Vortex dynamics in the cylinder wake, Ann. Rev. Fluid Mech. 28 (1996) 477–539.
[37] T. Ye, R. Mittal, H.S. Udaykumar, W. Shyy, An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundary, J. Comput. Phys. 156 (1999) 209–240.